TiMem: Temporal-Hierarchical Memory Consolidation for Long-Horizon Conversational Agents

Anonymous ACL submission

Abstract

Long-horizon conversational agents have to manage ever-growing interaction histories that quickly exceed the finite context windows of large language models (LLMs). Existing memory frameworks provide limited support for temporally structured information across hierarchical levels, often leading to fragmented memories and unstable long-horizon personalization. We present **TiMem**, a temporal–hierarchical memory framework that organizes conversations through a Temporal Memory Tree (TMT), enabling systematic memory consolidation from raw conversational observations to progressively abstracted persona representations. TiMem is characterized by three core properties: (1) temporal–hierarchical organization through TMT; (2) semantic-guided consolidation that enables memory integration across hierarchical levels without fine-tuning; and (3) complexity-aware memory recall that balances precision and efficiency across queries of varying complexity. Under a consistent evaluation setup, TiMem achieves state-of-the-art accuracy on both benchmarks, reaching 75.30% on LoCoMo and 76.88% on LongMemEval-S. It outperforms all evaluated baselines while reducing the recalled memory length by 52.20% on LoCoMo. Manifold analysis indicates clear persona separation on LoCoMo and reduced dispersion on LongMemEval-S. Overall, TiMem treats temporal continuity as a first-class organizing principle for long-horizon memory in conversational agents. Code is available at `https://anonymous.4open.science/r/TiMem`.

1 Introduction

Large Language Models (LLMs) have enabled conversational agents to evolve from short-horizon task solvers (Qian et al., 2024; Zeng et al., 2024; Zhang et al., 2024) to long-horizon personalized companions (Chen et al., 2024a; Li et al., 2025a; Zhang et al., 2025). Supporting such interactions requires

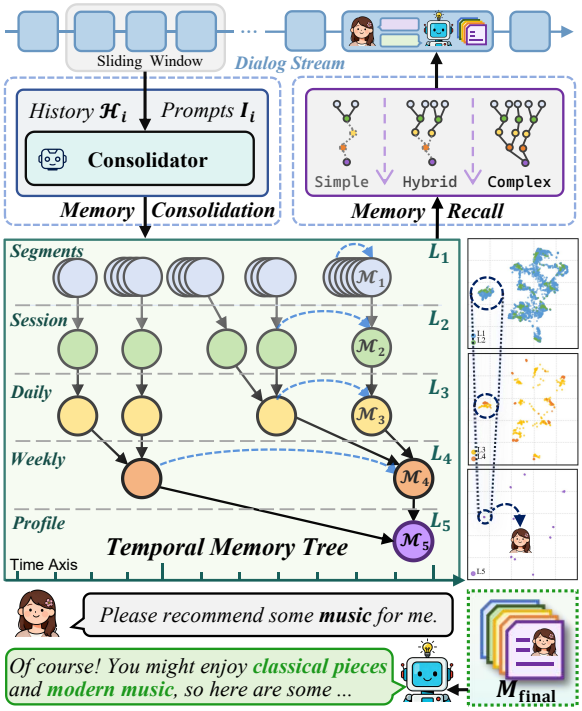

Figure 1: TiMem framework overview. The framework organizes conversational streams through a five-level TMT, consolidating memories from factual segments to persona profiles, with adaptive memory recall guided by query complexity.

two capabilities: maintaining temporal coherence as user states evolve, and forming stable representations by distilling consistent personas from dynamic experiences. However, interaction histories grow unbounded, while LLMs operate under finite context windows, making it harder to sustain temporally consistent personalization at scale. The key challenge is to transform long-horizon experience into compact representations that remain temporally grounded and useful for subsequent tasks.

Existing solutions under-emphasize temporal structure as a first-class constraint, and often lack explicit temporal containment guarantees across hierarchical levels. Parametric approaches expand context windows (Chen et al., 2023; et al., 2024)

or optimize internal context capacity (Bini et al., 2025; Bui et al., 2025), but they remain bounded by model architecture and do not provide persistent cross-session storage. External memory systems (Chhikara et al., 2025; Packer et al., 2024; Zhong et al., 2023) enable persistence but often rely on semantic similarity-driven clustering (Sarthi et al., 2024) or learned routing policies (Du et al., 2025), treating temporal structure as auxiliary metadata. As a result, memories from different periods can be aggregated without clear temporal boundaries, and retrieval may surface temporally distant evidence without an explicit ordering. For evolving users, persona modeling benefits from a time-ordered evidence chain rather than only semantically similar fragments.

Cognitive neuroscience provides a principled perspective on this problem. Human memory relies on complementary learning systems (McClelland et al., 1995), where **memory consolidation** (Squire et al., 2015) progressively transforms rapid episodic encoding into more stable semantic structures (Cowan et al., 2021). This adaptive process prioritizes goal-relevant information over indiscriminate retention. Translating this view to long-horizon agents suggests two design requirements: time should be encoded as an explicit structural constraint, and memory should be consolidated progressively across temporal granularities.

To this end, we introduce **TiMem**, a memory framework that uses temporal structure as the primary organizing principle and operationalizes consolidation in a computational form. TiMem consolidates fine-grained episodic interactions into higher-level semantic patterns and persona representations, rather than maintaining raw context buffers.

As illustrated in Figure 1, TiMem implements a hierarchical consolidation mechanism with three components and requires no additional fine-tuning in our experiments. (1) The **Temporal Memory Tree (TMT)** organizes memories with explicit temporal containment and order through tree constraints. (2) The **Memory Consolidator** performs instruction-guided consolidation; level-specific prompts control the abstraction level, enabling plug-and-play use across different LLM backends. (3) **Memory Recall** performs complexity-aware hierarchical retrieval: a recall planner selects appropriate hierarchy levels based on query complexity, and a recall gating step filters candidates to balance factual detail with higher-level personalization.

Our contributions are threefold: (1) the TMT, a novel structure that enforces explicit temporal containment and granularity for memory organization; (2) the TiMem framework, a temporal–hierarchical memory consolidation framework based on instruction-guided reasoning and complexity-adaptive recall, requiring no fine-tuning; (3) a comprehensive evaluation demonstrating TiMem's state-of-the-art accuracy (75.30% on LoCoMo, 76.88% on LongMemEval-S) and efficiency (52.20% reduced recalled context on LoCoMo), with ablations and manifold analyses providing insights into its hierarchical representations.

2 Related Work

Parametric Memory Approaches. Context window expansion methods such as Gemini (et al., 2024), LongLoRA (Chen et al., 2024b), and RoPE scaling (Chen et al., 2023) alleviate sequence length limits but incur quadratic computational costs and attention dilution (Liu et al., 2024). Parametric optimization approaches, including MemLoRA (Bini et al., 2025), HMT (He et al., 2025), and TRIM-KV (Bui et al., 2025), compress memory through adapter distillation or learned token retention. However, they remain constrained by architectural context windows and do not support persistent cross-session memory.

External Memory Management. Semantic clustering approaches, including Mem0 (Chhikara et al., 2025), RAPTOR (Sarthi et al., 2024), and MemTree (Rezazadeh et al., 2025), organize memory through embedding-based similarity aggregation. Graph-based approaches, including Zep (Rasmussen et al., 2025), LiCoMemory (Huang et al., 2025), and Theanine (iunn Ong et al., 2025), explicitly model entity relations and temporal knowledge. Cognitively motivated frameworks such as A-MEM (Xu et al., 2025), Nemori (Nan et al., 2025), ENGRAM (Patel and Patel, 2025), and RMM (Tan et al., 2025) employ self-organizing or agentic mechanisms, while preference-aware systems like MemoryBank (Zhong et al., 2023) and PAMU (Sun et al., 2025) support personalization through dynamic updates. OS-inspired memory systems such as MemGPT (Packer et al., 2024), MemoryOS (Kang et al., 2025), and MemOS (Li et al., 2025b) manage long contexts via hierarchical tiers and virtual memory mechanisms. However, most existing approaches do not treat temporal structure as a first-class organizing principle, resulting in fragmented memory representations and unstable long-horizon behavior.

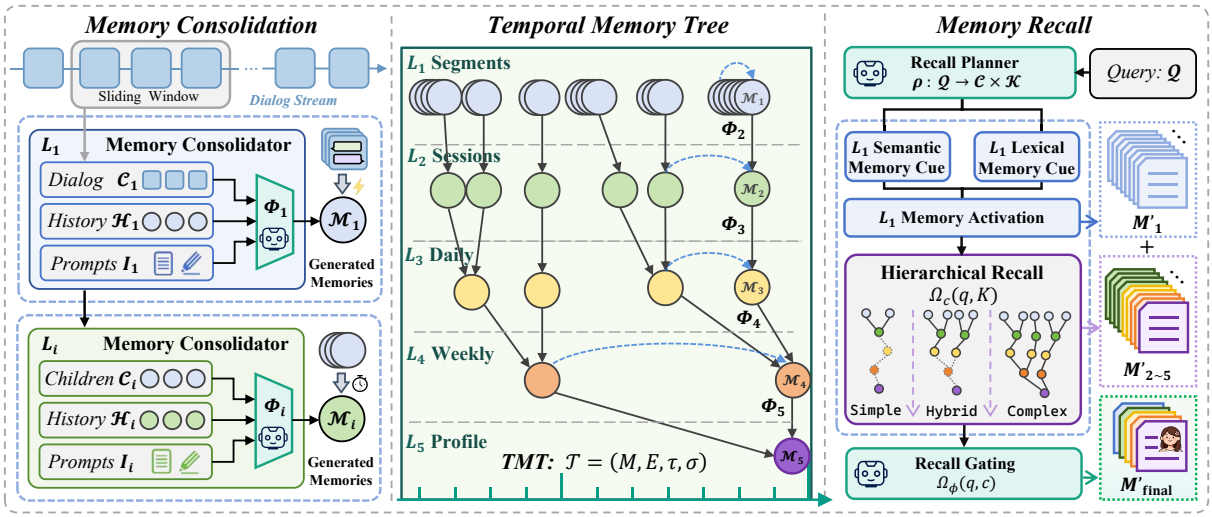

Figure 2: TiMem architecture overview: a five-layer TMT from level 1 segments to level 5 profiles, with a consolidation pipeline processing dialog into temporal-hierarchical memories, and a recall pipeline without fine-tuning, and includes a recall planner, hierarchical recall, and a recall gating module.

3 Methodology

We present **TiMem**, a temporal–hierarchical memory framework for long-horizon conversational agents. TiMem consists of (i) a TMT that encodes temporal structure, (ii) a Memory Consolidator that performs level-specific consolidation via instruction prompting without fine-tuning, and (iii) a Recall pipeline that uses a planner to select relevant memory levels and a recall gating module to retain query-relevant memories, as illustrated in Figure 2.

3.1 Temporal Memory Tree

The TMT provides a stable backbone for long-horizon memory: it preserves temporal coherence, supports progressive consolidation, and reduces noise by transforming details into higher-level abstracts. Lower-level memories cover short intervals and keep concrete details, while higher-level ones span longer intervals and store more consolidated representations. Each node m stores a time interval $\tau(m)$ and a semantic memory $\sigma(m)$. We use $\ell(m) \in \{1, \ldots, L\}$ to denote the level of node m, from fine-grained to generalized.

Definition. TMT is a hierarchical memory structure $\mathcal{T} = (M, E, \tau, \sigma)$ defined by:

- $M = \bigcup_{i=1}^{L} M_i$ is the set of memory nodes partitioned across L abstraction levels;
- $E \subseteq M \times M$ defines parent–child relationships where $\ell(m_u) = \ell(m_v) + 1, \forall (m_u, m_v) \in E$;
- τ assigns each node a temporal interval $\tau(m) = [t_{\text{start}}, t_{\text{end}}]$ which is continuous over periods;
- σ maps each node to a semantic memory $\sigma(m)$ stored as text and embeddings.

Structural Properties. The structure is governed by three principles that make temporal order explicit and enable progressive abstraction:

- **Temporal Containment**: $\tau(m_u) \supseteq \tau(m_v)$, for each parent-child edge $(m_u, m_v) \in E$, the parent interval covers the child interval.
- **Progressive Consolidation**: $|M_i| \leq |M_{i-1}|$ ensures higher-level memories are fewer, reflecting consolidation from fine-grained facts to patterns and profiles.
- **Semantic Consolidation**: Specified by level-specific instruction prompts $\mathcal{I}_i$, $\sigma(m_u) = \text{LLM}(\{\sigma(m_v)\}, \mathcal{I}_i)$ enables hierarchy specialization through the consolidation process.

Implementation TMT supports arbitrary L and τ configurations. For reproducibility, **TiMem uses a five-level hierarchy** (segment, session, day, week, profile). Each level performs a different type of consolidation, specified by level-specific instruction prompts $\mathcal{I}_i$:

- **Factual Summarization**: Segments L_1 distill key dialog details; Sessions L_2 merge into non-redundant event summaries.
- **Evolving Patterns**: Daily L_3 captures routine contexts and recurrent interests; Weekly L_4 integrates evolving behavioral features and preference patterns.
- **Persona Representation**: Profile L_5 is an incrementally refined profile capturing stable personality, preferences, and values from long-term patterns, updated on monthly intervals.

The framework is designed to be model-independent and does not require fine-tuning; it can be applied across different LLM backbones.

3.2 Memory Consolidation

TiMem constructs the hierarchy with a Memory Consolidator that converts dialog into structured memories and uses Stratified Scheduling to balance consolidation efficiency and computational cost.

3.2.1 Memory Consolidator

At level i, the consolidator generates new memories by prompting an LLM with (i) child memories, (ii) historical memories, and (iii) instruction prompts.

$$\Phi_i : \mathcal{C}_i \times \mathcal{H}_i \times \mathcal{I}_i \rightarrow M_i \tag{1}$$

In formula (1), $\mathcal{C}_i$ are child memories from level $i-1$, $\mathcal{H}_i$ provides short same-level history for continuity, and $\mathcal{I}_i$ are instruction prompts. We use $\mathcal{I}_1$-$\mathcal{I}_2$ for factual consolidation, $\mathcal{I}_3$-$\mathcal{I}_4$ for pattern consolidation, and $\mathcal{I}_5$ for profile representation. Example consolidator prompt is shown in Appendix E.3.1.

Child Memories We group the conversation timeline into intervals $g \in \mathcal{G}_i$ (e.g., sessions, days). For $i \geq 2$, child memories for each group are the lower-level nodes whose time spans fall inside g:

$$\mathcal{C}_i(g) = \{m \in M_{i-1} : \tau(m) \subseteq g\}, \quad i \geq 2 \tag{2}$$

At the base level (L_1), child memories are the raw dialog turns within the interval.

Historical Memories $\mathcal{H}_i$ consists of the w_i most recent memories from the same level M_i:

$$\mathcal{H}_i = \{m_{-j}^{(i)} : 1 \leq j \leq w_i\} \tag{3}$$

where $m_{-j}^{(i)}$ denotes the j-th most recent memory at level i. This sliding window provides continuity across temporal groups. We set $w_i = 3$ across all levels to ensure consolidation consistency.

3.2.2 Stratified Scheduling

Memory consolidation follows a two-tier scheduling strategy that balances freshness and efficiency:

- **Online consolidation (L_1):** Factual segment memories $m_k^{(1)}$ are generated immediately as the dialog progresses. With $w_d = 1$ dialog turn (one user–assistant exchange), the consolidator Φ_1 is invoked after each new turn to capture fine-grained evidence.
- **Scheduled consolidation (L_2-L_5):** Higher-level memories $m^{(i)}(g)$ are generated automatically when their temporal windows end. Upon closure of temporal group $g \in \mathcal{G}_i$, the framework triggers $\Phi_i(\mathcal{C}_i(g), \mathcal{H}_i(g); \mathcal{I}_i)$ to consolidate child memories into a higher-level, more abstract representation.

Thus, this stratified design ensures that factual details are captured in real time while higher-level consolidation is aligned with predefined temporal boundaries.

3.3 Memory Recall

Memory recall traverses the TMT to surface relevant memories, balancing precision, efficiency, and context length. It adapts scope to query complexity: simple questions target exact evidence, while complex ones incorporate higher-level memories to provide long-range context. A final recall gating step filters redundancy and conflicts, retaining only memories required for the current interaction.

3.3.1 Recall Planner

The planner $p : \mathcal{Q} \rightarrow \mathcal{C} \times \mathcal{K}$ maps a query q to a complexity label $c \in \{\text{simple}, \text{hybrid}, \text{complex}\}$ and keywords K. We obtain both by prompting an LLM (Appendix E.1.1), without dataset-specific training or labeled annotations.

Query complexity determines which TMT levels to search. We define three layer groups:

- **Factual Layers** ($\mathcal{L}_{\text{fact}}$): L_1-L_2 capturing fine-grained event details.
- **Pattern Layers** ($\mathcal{L}_{\text{patt}}$): L_3-L_4 behavioral trends and patterns.
- **Profile Layer** ($\mathcal{L}_{\text{prof}}$): L_5 synthesizing long-term, stable characteristics.

Although *simple* queries are short, they can still ask about stable preferences, so we include the profile layer L_5 by default. Intermediate pattern layers are useful for cross-event reasoning and are therefore emphasized in *hybrid* and *complex* queries.

The recall strategy $\mathcal{S}$ maps complexity c to subsets of TMT:

$$\mathcal{S}(\text{simple}) = \mathcal{L}_{\text{fact}} \cup \mathcal{L}_{\text{prof}} \tag{4}$$

$$\mathcal{S}(\text{hybrid}) = \mathcal{L}_{\text{fact}} \cup \mathcal{L}_{\text{patt}}^{\text{partial}} \cup \mathcal{L}_{\text{prof}} \tag{5}$$

$$\mathcal{S}(\text{complex}) = \mathcal{L}_{\text{fact}} \cup \mathcal{L}_{\text{patt}} \cup \mathcal{L}_{\text{prof}} \tag{6}$$

where $\mathcal{L}_{\text{patt}}^{\text{partial}}$ recalls L_3 memories, while $\mathcal{L}_{\text{patt}}$ recalls more L_3 and L_4 memories. Simple queries bypass intermediate consolidated memories by directly accessing factual details and stable profiles, while complex ones traverse the full hierarchy to capture information at all levels.

3.3.2 Hierarchical Recall

Hierarchical recall operates in two stages: leaf selection at the base level, followed by hierarchical recall propagation through memory subtrees.

Stage 1: Base-Level Memory Activation At L_1, dual-channel scoring combines semantic similarity and lexical matching through fusion:

$$s(m, q, K) = \lambda s_{\text{sem}}(m, q) + (1 - \lambda) s_{\text{lex}}(m, K) \quad (7)$$

where s_{sem} is cosine similarity between embeddings, s_{lex} is BM25 score for keyword matching, and $\lambda \in [0, 1]$ balances both channels. The top-k_1 scoring segments form the leaf set $\Omega_1(q, K)$.

Stage 2: Hierarchical Recall Propagation For each leaf $m \in \Omega_1$, we collect its ancestors at the hierarchy levels selected by $\mathcal{S}(c)$:

$$\mathcal{A}(m, c) = \{m' \in M : m \preceq m',\ \ell(m') \in \mathcal{S}(c)\} \quad (8)$$

where $m \preceq m'$ denotes that m' is an ancestor of m, and $\mathcal{S}(c)$ restricts recall to levels specified by complexity c. The complete candidate set integrates leaves and their ancestors:

$$\Omega_c(q, K) = \Omega_1(q, K) \cup \bigcup_{m \in \Omega_1(q,K)} \mathcal{A}(m, c) \quad (9)$$

For brevity, we denote this candidate set as Ω_c. The number of recalled memories per level is determined by query complexity; specific configurations are detailed in Appendix B.3.

3.3.3 Recall Gating

Recall gating implements **recall-time forgetting**: after collecting candidate memories, we keep only the truly useful ones for answering the query.

The recall gating module ϕ receives query q, its complexity c, and the candidate set Ω_c organized by hierarchy levels. It prompts an LLM to determine whether each memory should be retained:

$$\Omega_\phi(q, c) = \{m \in \Omega_c \mid \phi(m, q, c) = \text{retain}\} \quad (10)$$

where $\phi(m, q, c)$ denotes the LLM's retention decision for memory m given query q and complexity c. Query complexity guides the breadth of retention: simple queries favor precision by retaining fewer memories, while complex queries favor recall by accepting broader context. Example recall gating prompt template is in Appendix E.3.2.

The retained memories are ranked by hierarchy level and temporal proximity within each level:

$$\Omega_{\text{final}}(q, c) = \text{sort}\big(\Omega_\phi(q, c),\ \text{key}{=}(\ell(m), |t_q - t_m|)\big) \quad (11)$$

where $\ell(m)$ denotes the hierarchy level, t_q is the query time, and $t_m = t_{\text{end}}(m)$ so $|t_q - t_m|$ measures temporal distance, organizing relevant memories by recency within each consolidation level, thereby ensuring concise, temporally coherent, and information-dense responses.

3.3.4 Recall Pipeline

The complete recall integrates three stages:

1. **Recall planner**: $p(q) \rightarrow (c, K)$ predicts complexity c and extracts keywords K to determine the hierarchical search scope.
2. **Hierarchical Recall**: Dual-channel scoring selects L_1 leaves, then hierarchical recall propagation collects relevant ancestors at planner-specified levels, forming the candidate set Ω_c.
3. **Recall Gating**: The refiner filters the candidates based on query relevance and temporal consistency, then orders them to produce the final memory set $\Omega_{\text{final}}(q, c)$.

This pipeline enables complexity-adaptive recall that balances precision and temporal relevance across TMT's hierarchical structure.

4 Experiments

4.1 Experimental Setup

Datasets We evaluate on two long-term conversational memory benchmarks: **LoCoMo** (Maharana et al., 2024), a dataset with 10 user groups across multi-session dialog, and **LongMemEval-S** (Wu et al., 2025), including 500 conversations designed for very-long memory processing evaluation.

Baselines We compare TiMem with five representative memory baselines using their recommended configurations: **MemoryBank** (Zhong et al., 2023), **Mem0** (Chhikara et al., 2025), **A-MEM** (Xu et al., 2025), **MemoryOS** (Kang et al., 2025), and **MemOS** (Li et al., 2025b).

Implementation Details For fair comparison, all methods use the same LLM and embedding setup: `gpt-4o-mini-2024-07-18` for generation and recall, `Qwen3-Embedding-0.6B` for embeddings, and recall budget $k = 20$. TiMem uses $\lambda = 0.9$ and $w_i = 3$. We use the LLM-as-a-Judge (LLJ), where an LLM judges answer correctness; we report accuracy along with memory tokens and recall latency for efficiency. Details are in Appendix B.

4.2 Main Results

4.2.1 Results on LoCoMo

Table 1 shows that TiMem achieves the best overall LLJ accuracy on LoCoMo at 75.30% ± 0.16%. It outperforms the strongest evaluated baseline, MemOS, at 69.24% ± 0.11%. TiMem also improves F1 and ROUGE-L (RL) to 54.40 and 54.68 in percentage, and achieves the best LLJ score in each question type. We compute LLJ using Mem0's evaluation prompt template, as shown in Appendix E.1.2.

Method	Single-Hop ↑ LLJ (841Q)	Temporal ↑ LLJ (321Q)	Open-Domain ↑ LLJ (96Q)	Multi-Hop ↑ LLJ (282Q)	Overall ↑ LLJ (1540Q)	F1 ↑	RL ↑
MemoryBank	46.18 ± 0.32	29.34 ± 0.45	36.67 ± 0.47	33.36 ± 0.54	39.77 ± 0.27	25.78	25.15
A-MEM	52.82 ± 0.28	60.87 ± 0.37	43.75 ± 0.00	38.37 ± 0.27	51.29 ± 0.06	30.37	36.85
Mem0	62.09 ± 0.42	59.25 ± 0.41	37.70 ± 0.47	50.14 ± 0.32	57.79 ± 0.34	42.52	44.14
MemoryOS	68.37 ± 0.46	52.46 ± 0.49	46.67 ± 1.79	52.76 ± 0.49	60.79 ± 0.48	45.36	43.74
MemOS	76.07 ± 0.10	69.47 ± 0.25	45.14 ± 0.49	56.85 ± 0.69	69.24 ± 0.11	45.02	47.41
TiMem (Ours)	**81.43 ± 0.05**	**77.63 ± 0.34**	**52.08 ± 0.74**	**62.20 ± 0.82**	**75.30 ± 0.16**	**54.40**	**54.68**

Table 1: Performance comparison on LoCoMo benchmark. Categories include Single-Hop, Temporal, Open-Domain, and Multi-Hop. Best results are **bolded**; underline indicates second best.

Method	LongMemEval-S Task Categories KU ↑ (78Q)	MS ↑ (133Q)	SSA ↑ (56Q)	SSP ↑ (30Q)	SSU ↑ (70Q)	TR ↑ (133Q)	Overall ↑ (500Q)
Answer Model: GPT-4o-mini-2024-07-18							
MemoryBank	21.79 ± 0.00	9.77 ± 0.00	50.00 ± 0.00	12.00 ± 1.83	29.71 ± 0.64	17.14 ± 0.34	21.04 ± 0.09
A-MEM	72.82 ± 0.51	40.30 ± 0.37	**87.50 ± 0.00**	39.33 ± 2.49	82.86 ± 0.00	36.09 ± 0.48	55.44 ± 0.15
Mem0	78.72 ± 0.70	66.17 ± 0.92	51.79 ± 0.00	50.00 ± 2.36	94.29 ± 0.00	49.17 ± 0.67	64.96 ± 0.41
MemoryOS	56.15 ± 0.57	44.81 ± 0.41	78.18 ± 0.00	51.33 ± 1.83	81.14 ± 0.64	53.38 ± 0.00	58.04 ± 0.18
MemOS	76.67 ± 0.51	58.80 ± 0.30	67.86 ± 0.00	50.67 ± 1.33	93.71 ± 0.70	65.11 ± 0.37	68.68 ± 0.16
TiMem (Ours)	**86.16 ± 1.07**	**70.83 ± 0.98**	82.14 ± 0.00	**63.33 ± 0.00**	**95.71 ± 0.00**	**68.42 ± 0.00**	**76.88 ± 0.30**
Answer Model: GPT-4o-2024-11-20							
MemoryBank	22.56 ± 0.70	12.78 ± 0.00	61.43 ± 0.98	13.33 ± 0.00	33.43 ± 0.78	13.53 ± 0.00	22.88 ± 0.23
A-MEM	87.18 ± 0.00	45.26 ± 0.30	83.21 ± 0.87	56.67 ± 2.98	90.00 ± 0.00	46.77 ± 0.30	63.40 ± 0.33
Mem0	84.87 ± 0.57	65.11 ± 0.41	55.00 ± 0.80	60.67 ± 1.49	95.71 ± 0.00	51.88 ± 0.00	67.56 ± 0.30
MemoryOS	60.00 ± 0.57	51.13 ± 0.53	80.00 ± 0.00	53.33 ± 0.00	82.86 ± 0.00	54.59 ± 0.67	61.20 ± 0.23
MemOS	76.07 ± 0.60	68.42 ± 0.00	63.69 ± 0.84	**64.44 ± 1.57**	92.86 ± 0.00	71.43 ± 0.61	73.07 ± 0.25
TiMem (Ours)	**87.69 ± 0.70**	**72.78 ± 0.34**	**85.71 ± 0.00**	55.33 ± 1.83	**96.28 ± 0.78**	**73.38 ± 1.14**	**78.96 ± 0.26**

Table 2: Performance comparison on the LongMemEval-S benchmark, reporting LLJ accuracy by task type. Categories include KU: Knowledge Update, MS: Multi-Session, SSA/P/U: Single-Session Assistant/Preference/User, and TR: Temporal Reasoning. Best results are **bolded**; underline indicates second best.

4.2.2 Results on LongMemEval-S

Table 2 shows that TiMem achieves the best overall LLJ accuracy on LongMemEval-S at 76.88% ± 0.30% with `gpt-4o-mini-2024-07-18` as the answer model, outperforming the evaluated baselines. With `gpt-4o-2024-11-20` as the answer model, TiMem remains best overall at 78.96% ± 0.26%. The QA and LLJ protocol follow the official LongMemEval-S evaluation template, as shown in Appendix E.2.1 and E.2.2.

4.3 Ablation Studies

We ablate TiMem to isolate the contribution of its main components. All ablations use `gpt-4o-mini-2024-07-18` for LLM operations and `Qwen3-Embedding-0.6B` for embeddings.

4.3.1 Planner and Recall Gating

Table 3 compares seven configurations of recall scope and gating. Fixed-scope recall under-recalls for Simple queries and introduces noise for Complex ones. Recall gating sharply reduces memory length—for example, from 3710.30 to 367.68 tokens on LoCoMo under Simple—but accuracy drops when the scope is overly narrow. Among fixed-scope settings, Hybrid + Recall Gating performs best, achieving 73.38% on LoCoMo and 75.00% on LongMemEval-S. The adaptive planner further improves the accuracy–cost trade-off, reaching 75.30% with 511.25 tokens on LoCoMo and 76.88% with 1270.62 tokens on LongMemEval-S.

4.3.2 Hierarchical Architecture

Table 4 examines hierarchy depth and recall strategy. With L1-only memories, hierarchical recall propagation raises LongMemEval-S LLJ from 57.40% to 72.40% compared to flat recall, indicating that hierarchical propagation recovers necessary temporal dependencies. However, L1-only remains below the full hierarchy on LoCoMo, as isolated factual fragments often lack the broader context required for complex queries. Using only high-level layers (L2–L5) further reduces accuracy, confirming that summaries alone cannot replace fine-grained evidence. Overall, the full hierarchy combines precise L1 grounding with contextual

Configuration	LoCoMo		LongMemEval-S	
	LLJ ↑	Mem Len ↓	LLJ ↑	Mem Len ↓
w/o Planner, w/o Gating				
- Simple	73.51	3710.30	73.20	3371.53
- Hybrid	72.40	4376.40	74.00	4054.78
- Complex	72.86	5658.26	74.40	5685.68
w/o Planner, w Gating				
- Simple	71.88	**367.68**	69.00	**397.04**
- Hybrid	73.38	691.59	75.00	1673.93
- Complex	72.92	4479.06	74.20	3028.68
w Planner, w/o Gating				
- Planned	72.99	4411.09	73.80	3941.98
w P., w G. (Baseline)	**75.30**	511.25	**76.88**	1270.62

Table 3: Recall Planner and Recall Gating effectiveness. The planned configuration of the TiMem baseline achieves the best balance between accuracy and memory length.

Memory Layers	LoCoMo		LongMemEval-S	
	LLJ ↑	Mem Len ↓	LLJ ↑	Mem Len ↓
L1 only (base layer)				
w Flat Rec.	70.06	995.15	57.40	1823.98
w Hier. Rec.	73.18	**361.23**	72.40	**437.42**
L2-L5 only (high-level)				
w Flat Rec.	51.23	2348.49	48.00	2657.68
w Hier. Rec.	57.08	3786.44	64.20	2344.92
L1-L5 (full hierarchy)				
w Flat Rec.	70.71	1715.65	55.40	4519.26
w H. R. (Baseline)	**75.30**	511.25	**76.88**	1270.62

Table 4: Hierarchical vs. Flat Architectures. Comparison of L1-only, L2-L5 only, and Full Hierarchy with flat vs. hierarchical recall strategies.

understanding from L2–L5, achieving the best performance on both datasets.

These ablations support TiMem's core design: the temporal hierarchy provides both factual precision and contextual understanding through memory consolidation, while the adaptive planner dynamically balances recall scope.

4.4 Memory Manifold Analysis

Figure 3 illustrates UMAP visualization of TiMem memory embeddings on LoCoMo and LongMemEval-S through different hierarchies. It shows that consolidation reshapes memory geometry differently across datasets. On LoCoMo, higher-level memories separate users more clearly, with clustering quality improving 6.2×, indicating effective persona feature distillation. On LongMemEval-S, consolidation reduces spatial dispersion by 50%, suggesting suppression of sampling noise while retaining core persona attributes. These complementary behaviors suggest that TiMem preserves semantically salient patterns beyond uniform averaging. Detailed metrics are in Appendix D.

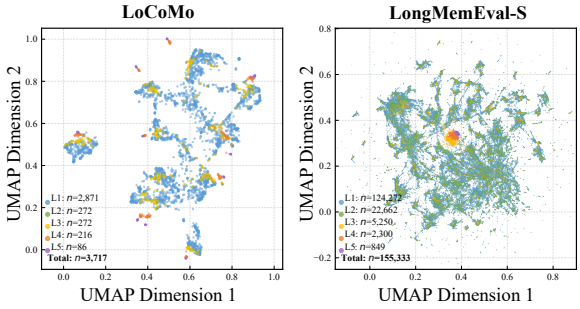

Figure 3: UMAP visualization of memory embeddings. **Left:** LoCoMo exhibits 10 user groups separation through hierarchical consolidation. **Right:** LongMemEval-S converges toward shared persona structure through noise suppression.

4.5 Recall Efficiency Analysis

Table 5 reports recall efficiency metrics: memory context length and latency.

Method	LoCoMo		LongMemEval-S	
	Memory ↓ (tokens)	Latency ↓ (P50/P95)	Memory ↓ (tokens)	Latency ↓ (P50/P95)
MemoryBank	8063.77	9.46/13.07	13906.81	10.74/14.50
A-MEM	2431.4	1.74/7.23	3971.6	5.12/11.89
Mem0	1070.10	2.44/4.29	1647.56	3.64/6.11
MemoryOS	4659.09	**1.66/2.21**	7574.30	**1.63**/3.95
MemOS	1371.42	1.69/3.44	**1091.51**	1.64/**2.70**
TiMem (Ours)	**511.25**	2.35/4.91	1270.62	1.76/4.48

Table 5: **Recall efficiency metrics.** Memory context length and P50/P95 latency across benchmarks. TiMem significantly reduces context load compared to baselines while maintaining low latency.

On LoCoMo, TiMem recalls 511.25 tokens per query versus 1,070.10 for Mem0, reducing context length by 52.20%. P50 recall latency is 2.35s on LoCoMo and 1.76s on LongMemEval-S. Context length increases with query complexity, and LongMemEval-S queries are more diverse and recall more context. Latency includes planner, recall, and gating, with LLM calls as the dominant.

4.6 Parameter Studies

We conduct a parameter study on LoCoMo.

LLM Configuration Under the same answering and judgement protocol, TiMem is portable across internal LLMs for memory operations. End-to-end performance is primarily driven by the answering LLM, with the best configuration reaching 80.45%, indicating that answer-time reasoning dominates once memory quality is adequate.

Segment Granularity Increasing the L1 segment size consistently degrades accuracy, dropping from 75.30% at 1 turn to 65.26% at 8 turns, indicating that finer-grained segments better preserve atomic evidence for downstream QA.

Detailed experimental designs, results, and cross-configuration analysis are provided in Appendix C.

4.7 Case Study

Figure 4 contrasts TiMem's hierarchical consolidation against Mem0 fragmented memories.

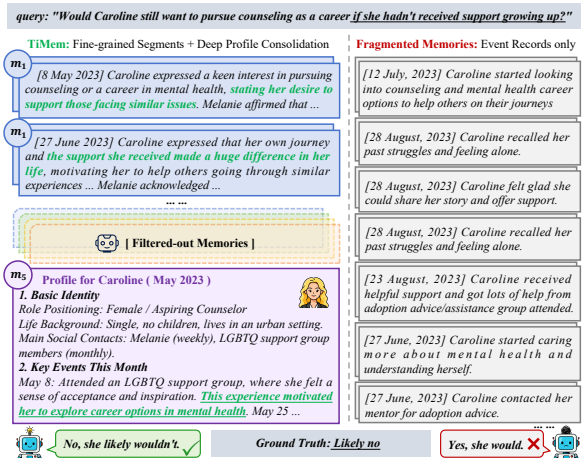

Figure 4: Case study comparing TiMem and a non-hierarchical baseline. TiMem's hierarchical consolidation organizes timestamped evidence into coherent chains and a persona profile, whereas the baseline recalls only isolated event records.

TiMem recalls segments establishing causal dependency, with the consolidated L5 profile connecting her career aspiration to formative experiences. The recall gating module excludes memories lacking true relevance. This structured causality yields the correct answer: ***No, she likely wouldn't***.

Mem0 as a representative baseline recalls fragmented factual memories. Without hierarchical consolidation, the framework fails to construct the *support→career chain*, producing an inverted answer: ***Yes, she would***.

This comparison highlights how TMT's temporal containment and instruction-based consolidation organize episodic evidence into a coherent inferential structure for counterfactual reasoning.

4.8 Discussion

Our experiments suggest three key takeaways for long-horizon memory in conversational agents.

Temporal continuity is an effective organizing principle. By enforcing temporal containment, TMT provides stable temporal leaves for consolidation and recall, instead of treating semantic similarity as the primary structure. Ablation studies and manifold analysis indicate that this temporal hierarchy enables effective compression: it facilitates the construction of temporal evidence chains, amplifies user-specific distinctions, and suppresses noise in long dialogs.

Semantic-guided consolidation makes abstraction explicit and portable. Level-specific prompts encourage distinct consolidation objectives across layers without architecture-specific tuning. Empirically, hierarchical consolidation outperforms the evaluated methods, indicating that progressive transformation over temporally grouped memories is beneficial beyond storing more text.

Recall reflects a practical trade-off between precision and efficiency. The complexity-aware recall planner consistently outperforms fixed recall scopes, while recall gating is most effective when the candidate set contains distractors. For highly complex queries, broader context may outweigh aggressive filtering, and planner errors can expand or shrink the recall scope; in practice, recall budgets can be tuned to different application needs.

Overall, TiMem suggests that combining temporal organization with hierarchical consolidation and adaptive recall yields compact yet grounded long-term memory for conversational agents.

5 Conclusion

We introduced **TiMem**, a temporal–hierarchical memory framework for long-horizon conversational agents, which treats temporal continuity as a first-class organizing principle for long-term memory personalization. TiMem provides: (i) the **TMT**, a structure that enforces temporal containment and order; (ii) **instruction-guided consolidation without fine-tuning** that progressively transforms raw dialog into higher-level patterns and incrementally refined profiles updated monthly; and (iii) **complexity-aware recall** that plans the recall scope, propagates evidence hierarchically from activated leaves, and applies recall-time gating to retain only query-relevant memories.

Under a consistent evaluation setup, TiMem achieves state-of-the-art accuracy of **75.30%** on **LoCoMo** and **76.88%** on **LongMemEval-S**, while **reducing recalled context by 52.20%** on LoCoMo via recall planning and gating. Manifold analysis indicates that temporal consolidation yields persona separation while reducing dispersion, supporting coherent long-horizon memory representations.

We view TiMem as a practical and interpretable foundation for long-term agent memory. Future directions include combining temporal hierarchies with richer structured memory representations and incorporating storage-time forgetting and adaptive temporal boundaries to further improve efficiency and robustness.

6 Limitations

LLM Middleware Performance Consolidation and recall modules rely on general-purpose LLMs through instruction prompts. Fine-tuning specialized smaller models for these operations may improve efficiency while maintaining module functionality.

Structured Representation High-level memories lack explicit categorical structures or knowledge graphs. Hybrid architectures combining temporal hierarchies with typed entity representations may better capture multi-dimensional content.

Forgetting Mechanism The framework lacks storage-time forgetting mechanism. Future work should explore effective storage-level forgetting methods that selectively consolidate memories while maintaining critical facts and recurring patterns, balancing efficiency with factual integrity.

Temporal Parameterization TiMem uses realistic temporal boundaries for reproducibility. Adaptive temporal boundaries detection or interaction-density scheduling could enhance domain transferability.

7 Ethics Statement

This work does not involve human subjects or personally identifiable information. Experiments use publicly available benchmarks under appropriate licenses. TiMem enforces strict user-group isolation by design: each memory tree is scoped to a single user, with no cross-user memory sharing or aggregation, protecting individual privacy. Deployed systems should implement secure storage, explicit user consent, and data deletion mechanisms. As with any LLM-based system, practitioners should monitor for potential biases in memory consolidation and ensure transparency about retention policies.

References

Massimo Bini, Ondrej Bohdal, Umberto Michieli, Zeynep Akata, Mete Ozay, and Taha Ceritli. 2025. Memlora: Distilling expert adapters for on-device memory systems. *Preprint*, arXiv:2512.04763.

Ngoc Bui, Shubham Sharma, Simran Lamba, Saumitra Mishra, and Rex Ying. 2025. Cache what lasts: Token retention for memory-bounded kv cache in llms. *Preprint*, arXiv:2512.03324.

Jiangjie Chen, Xintao Wang, Rui Xu, Siyu Yuan, Yikai Zhang, Wei Shi, Jian Xie, Shuang Li, Ruihan Yang, Tinghui Zhu, Aili Chen, Nianqi Li, Lida Chen, Caiyu Hu, Siye Wu, Scott Ren, Ziquan Fu, and Yanghua Xiao. 2024a. From persona to personalization: A survey on role-playing language agents. *Preprint*, arXiv:2404.18231.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *Preprint*, arXiv:2306.15595.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024b. Longlora: Efficient fine-tuning of long-context large language models. *Preprint*, arXiv:2309.12307.

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

Emily T Cowan, Anna C Schapiro, Joseph E Dunsmoor, and Vishnu P Murty. 2021. Memory consolidation as an adaptive process. *Psychonomic Bulletin & Review*, 28(6):1796–1810.

Xingbo Du, Loka Li, Duzhen Zhang, and Le Song. 2025. Memr^3: Memory retrieval via reflective reasoning for llm agents. *Preprint*, arXiv:2512.20237.

Gemini Team et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Zifan He, Zongyue Qin, Neha Prakriya, and Yizhou Sun. 2025. Hmt: Hierarchical memory transformer for efficient long context language processing. *Preprint*, arXiv:2405.06067.

Zhengjun Huang, Zhoujin Tian, Qintian Guo, Fangyuan Zhang, Yingli Zhou, Di Jiang, and Xiaofang Zhou. 2025. Licomemory: Lightweight and cognitive agentic memory for efficient long-term reasoning. *Preprint*, arXiv:2511.01448.

Kai Tzu iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seung won Hwang, Dongha Lee, and Jinyoung Yeo. 2025. Towards lifelong dialogue agents via timeline-based memory management. *Preprint*, arXiv:2406.10996.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. *Preprint*, arXiv:2506.06326.

Hao Li, Chenghao Yang, An Zhang, Yang Deng, Xiang Wang, and Tat-Seng Chua. 2025a. Hello again! LLM-powered personalized agent for long-term dialogue. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5259–5276, Albuquerque, New Mexico. Association for Computational Linguistics.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, Junpeng Ren, Zehao Lin, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhiqiang Yin, Qingchen Yu, Bo Tang, and 3 others. 2025b. Memos: An operating system for memory-augmented generation (mag) in large language models. *Preprint*, arXiv:2505.22101.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

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

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. 2025. Nemori: Self-organizing agent memory inspired by cognitive science. *Preprint*, arXiv:2508.03341.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Nishant Patel and Apurv Patel. 2025. Engram: Effective, lightweight memory orchestration for conversational agents. *Preprint*, arXiv:2511.12960.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.

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

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. 2025. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for llms. *Preprint*, arXiv:2410.14052.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *Preprint*, arXiv:2401.18059.

Larry R Squire, Lisa Genzel, John T Wixted, and Richard G Morris. 2015. Memory consolidation. *Cold Spring Harbor perspectives in biology*, 7(8):a021766.

Haoran Sun, Zekun Zhang, and Shaoning Zeng. 2025. Preference-aware memory update for long-term llm agents. *Preprint*, arXiv:2510.09720.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long T. Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand 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. *Preprint*, arXiv:2503.08026.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *Preprint*, arXiv:2410.10813.

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

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, Bangkok, Thailand. Association for Computational Linguistics.

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024. Android in the zoo: Chain-of-action-thought for GUI agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12016–12031, Miami, Florida, USA. Association for Computational Linguistics.

Zhehao Zhang, Ryan A. Rossi, Branislav Kveton, Yijia Shao, Diyi Yang, Hamed Zamani, Franck Dernoncourt, Joe Barrow, Tong Yu, Sungchul Kim, Ruiyi Zhang, Jiuxiang Gu, Tyler Derr, Hongjie Chen, Junda Wu, Xiang Chen, Zichao Wang, Subrata Mitra, Nedim Lipka, and 2 others. 2025. Personalization of large language models: A survey. *Preprint*, arXiv:2411.00027.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. Memorybank: Enhancing large language models with long-term memory. *Preprint*, arXiv:2305.10250.

A Dataset Details

A.1 LoCoMo Benchmark

LoCoMo (Maharana et al., 2024) consists of 10 user groups and multi-session conversations spanning over six months on average. Conversations are grounded in dialog streams with explicit timestamps. We use 1,540 questions in the benchmark covering (1) single-hop, (2) multi-hop, (3) open-domain, and (4) temporal reasoning.

A.2 LongMemEval-S Benchmark

LongMemEval-S (Wu et al., 2025) is a synthetic benchmark with 500 conversations and 500 questions. It simulates online memory processing and assesses five capabilities: (1) information extraction from single user/assistant turns, (2) multi-session reasoning, (3) knowledge updates, (4) temporal reasoning, and (5) abstention. The benchmark constructs 500 user personas by sampling attributes from a shared persona template pool.

B Implementation Details

B.1 Memory Consolidation Configuration

All frameworks use `gpt-4o-mini-2024-07-18` for memory consolidation with temperature 0.7 and `Qwen3-Embedding-0.6B` (dimension 1024) for embeddings, ensuring consistent processing capabilities across comparisons. For TiMem: L1 memories created online using non-overlapping sliding window with segment size $w_d = 1$ turn (one user–assistant exchange); L2–L5 generated through scheduled aggregation at temporal boundaries (session end, daily, weekly, and $\mathcal{G}_5$ monthly intervals for profiles). Historical context window $w_i = 3$ prior memories per layer, balancing continuity and computational cost.

B.2 Question Answering Configuration

Basic settings Baselines are evaluated using their recommended configurations with the following unified settings: `gpt-4o-mini-2024-07-18` for consolidation and recall, `Qwen3-Embedding-0.6B` for embeddings, and recall budget k=20 memories.
LoCoMo Question answering uses `gpt-4o-mini-2024-07-18` with Mem0's prompt templates.
LongMemEval-S We use a unified template to clearly separate roles. *Internal LLM* (consolidation/planner/gating) is `gpt-4o-mini-2024-07-18`. *External LLM* is `gpt-4o-mini-2024-07-18` by default (we additionally report results with `gpt-4o-2024-11-20`). *Judge LLM* follows the official LongMemEval-S evaluation prompt. The evaluation prompt was meta-evaluated by its authors to achieve >97% agreement with human experts, supporting the reliability of LLJ-based scoring on this benchmark (Wu et al., 2025).

B.3 Recall Configuration

TiMem's hierarchical recall uses complexity-aware configurations with two LLM calls per query: a **recall planner** (1 call) and **recall gating** (1 call).

Stage	Operation and Key Parameters
1. Recall Planner (1 LLM call)	
	Predicts complexity $c \in \{\text{simple}, \text{hybrid}, \text{complex}\}$ and extracts keywords K to set level-specific budgets and search scope $\mathcal{S}(c)$.
2. Hierarchical Recall (no LLM calls)	
Leaf Activation	Score L_1 leaves by $s(m, q, K) = \lambda s_{\text{sem}} + (1-\lambda)s_{\text{lex}}$ with λ=0.9 (cosine similarity + BM25), then select top-k_1=20.
Ancestor Collection	For each activated leaf, collect ancestors whose levels satisfy $\ell(m) \in \mathcal{S}(c)$ (deterministic traversal).
Budgeting	Keep up to: **Simple** (L_1:20, L_2:4, L_5:1); **Hybrid** (L_1:20, L_2:4, L_3:2, L_5:1); **Complex** (L_1:20, L_2:8, L_3:4, L_4:2, L_5:1).
Pruning / Early Stop	If candidates exceed per-level budgets, prune by similarity scores; if fewer ancestors exist, terminate early.
3. Recall Gating (1 LLM call)	
	Prompt an LLM to retain/drop each candidate memory conditioned on (q, c), producing the final memory set Ω_{final}.

Table 6: Recall configuration in TiMem, organized by the three major stages: recall planner, hierarchical recall, and recall gating.

C Parameter Studies

C.1 LLM Configuration Analysis

We investigate the interplay between internal LLMs (used for memory consolidation and recall) and external LLMs (used for question answering). This experiment examines the memory system's quality and downstream reasoning capability separately, analyzing how different model combinations affect end-to-end performance.

Experimental Design We test two internal LLM configurations:

- **GPT-4o-mini**: A representative commercial LLM used for memory consolidation and recall.
- **Qwen3-32B**: A representative open-source LLM used as a production-oriented alternative.

As shown in table 7, for each internal configuration, we evaluate five external LLMs for question answering: `gpt-4o-mini`, `gpt-4o`, `qwen3-8b`, `qwen3-32b`, and `qwen3-235b-a22b`. We report results from two LLM-as-judge evaluators: `gpt-4o-mini` (denoted LLJ-G) and `qwen3-32b` (denoted LLJ-Q), and observe consistent trends across both judges.

C.2 Segment Granularity Analysis

We analyze the impact of L1 segment size on memory quality and retrieval effectiveness. Segment

Answer Model	F1 ↑	RL ↑	LLJ-G ↑	LLJ-Q ↑
Internal: GPT-4o-mini-2024-07-18				
GPT-4o-mini	54.40	54.68	75.30	72.86
GPT-4o	**56.14**	**56.40**	74.03	72.60
Qwen3-8B	46.58	47.00	66.04	64.68
Qwen3-32B	50.45	50.82	74.61	73.38
Qwen3-235B-A22B	42.17	43.60	**80.45**	**79.03**
Internal: Qwen3-32B				
GPT-4o-mini	51.50	51.74	71.23	70.00
GPT-4o	**53.65**	**53.90**	72.60	70.19
Qwen3-8B	44.59	45.29	64.16	62.34
Qwen3-32B	46.74	47.15	74.74	72.73
Qwen3-235B-A22B	40.20	41.47	**77.73**	**76.23**

Table 7: LLM Configuration Analysis. Comparison of answering models across two internal memory generation models (GPT-4o-mini, Qwen3-32B).

granularity determines the trade-off between detail preservation and computational efficiency.

Experimental Design. We evaluate four segment sizes based on dialog turn counts: 1 turn (finest), 2 turns, 4 turns, and 8 turns (coarsest). All other parameters use default values.

Segment Size	F1	RL	LLJ (%)	Delta
1 turn (baseline)	54.40	54.68	75.30	–
2 turns	52.45	51.00	73.64	-1.66
4 turns	50.56	49.27	70.00	-5.30
8 turns	46.94	45.72	65.26	-10.04

Table 8: Impact of L1 segment granularity on LoCoMo performance. Segment size determines the number of dialogue turns aggregated into each L1 memory. Default configuration uses 1 turn. Delta shows performance change relative to 1-turn baseline.

As shown in table 8, the performance of the question-answering decreases as the size of the segment increases, indicating a trade-off between the size of the segment and the accuracy of the QA in practical applications.

D Memory Manifold Analysis

We analyze how hierarchical consolidation transforms memory structure using manifold metrics: Intrinsic Dimensionality, Silhouette Score, Spread, and Trustworthiness.

D.1 LoCoMo: Feature Distillation

Table 9 shows progressive user differentiation across hierarchy levels. L1 segments exhibit high dimensionality and low clustering quality, indicating that generic conversational patterns dominate at the segment level. Through hierarchical consolidation, dimensionality compresses by 5.6-fold to reach 13 dimensions at L5, while clustering quality improves by 6.2-fold to achieve 0.574 silhouette score. The separation ratio increases from 0.30 to 2.14, indicating extraction of user-specific features from dialog streams.

Layer	IntDim↓	Silh↑	Sep.Ratio↑	Trust↑	Cont↑
L1	73	0.093	0.30	0.950	0.941
L2	35	0.273	0.77	0.964	0.955
L3	37	0.274	0.76	0.963	0.955
L4	28	0.329	0.89	0.972	0.964
L5	13	**0.574**	**2.14**	0.818	0.862

Table 9: **LoCoMo manifold metrics.** Progressive feature separation from L1 to L5 evidenced by increasing Silhouette Score and Separation Ratio.

Layer	IntDim↓	Spread↓	CV↓	Radius95↓	Trust↑
L1	100	0.692	0.085	0.789	0.917
L2	100	0.672	0.078	0.761	0.942
L3	100	0.533	0.162	0.669	0.847
L4	82	0.432	0.180	0.575	0.812
L5	68	**0.345**	0.155	**0.444**	0.789

Table 10: **LongMemEval-S convergence metrics.** Reduced Spread and Radius95 indicate convergence toward unified persona templates from L1 to L5.

D.2 LongMemEval-S: Noise Suppression

Table 10 reveals convergence toward shared structure in the synthetic dataset. L1 exhibits high spread at 0.692 and maximum dimensionality at 100. Through consolidation, spread reduces by 50% to reach 0.345 at L5, while the effective radius (mean distance to centroid) shrinks from 0.789 to 0.444. Dimensionality remains saturated at 100 through L1-L4, then drops to 68 at L5, with the low-dimensional shared structure emerging through progressive consolidation.

D.3 Adaptive Consolidation

TiMem demonstrates adaptive consolidation that responds to different data characteristics. In LoCoMo conversations, the framework acts as a feature separator, increasing inter-user variance as separation ratio grows from 0.30 to 2.14. In synthetic LongMemEval-S data, it functions as a noise filter, reducing variance as spread decreases from 0.692 to 0.345. Both processes achieve semantic compression through dimensionality reduction, yet produce contrasting topological effects: expanding distinctiveness for diverse users versus contracting dispersion for noisy data. Trustworthiness scores exceeding 0.78 across all levels indicate that these manifold transformations preserve neighborhood relationships during dimensionality reduction.

E Prompt Templates

E.1 LoCoMo Benchmark

We adopt the prompt template from Mem0 (Chhikara et al., 2025) for LoCoMo QA and evaluation:

E.1.1 Question Answering Prompt

```
You are an intelligent memory assistant tasked with retrieving accurate information from
conversation memories.
# CONTEXT:
You have access to memories from two speakers in a conversation. These memories contain
timestamped information that may be relevant to answering the question.
# INSTRUCTIONS:
1. Carefully analyze all provided memories from both speakers
2. Pay special attention to the timestamps to determine the answer
3. If the question asks about a specific event or fact, look for direct evidence in the memories
4. If the memories contain contradictory information, prioritize the most recent memory
5. If there is a question about time references (like "last year", "two months ago", etc.),
calculate the actual date based on the memory timestamp. For example, if a memory from 4 May 2022
mentions "went to India last year," then the trip occurred in 2021.
6. Always convert relative time references to specific dates, months, or years. For example,
convert "last year" to "2022" or "two months ago" to "March 2023" based on the memory timestamp.
Ignore the reference while answering the question.
7. Focus only on the content of the memories from both speakers. Do not confuse character names
mentioned in memories with the actual users who created those memories.
8. The answer should be less than 5-6 words.
# APPROACH (Think step by step):
1. First, examine all memories that contain information related to the question
2. Examine the timestamps and content of these memories carefully
3. Look for explicit mentions of dates, times, locations, or events that answer the question
4. If the answer requires calculation (e.g., converting relative time references), show your work
5. Formulate a precise, concise answer based solely on the evidence in the memories
6. Double-check that your answer directly addresses the question asked
7. Ensure your final answer is specific and avoids vague time references
Relevant Memories:
{context_memories}
Question: {question}
Answer:
```

E.1.2 LLM-as-Judge Evaluation Prompt

```
Your task is to label an answer to a question as 'CORRECT' or 'WRONG'. You will be given the
following data:
  (1) a question (posed by one user to another user),
  (2) a 'gold' (ground truth) answer,
  (3) a generated answer
which you will score as CORRECT/WRONG.
The point of the question is to ask about something one user should know about the other user
based on their prior conversations. The gold answer will usually be a concise and short answer
that includes the referenced topic, for example:
Question: Do you remember what I got the last time I went to Hawaii?
Gold answer: A shell necklace
The generated answer might be much longer, but you should be generous with your grading - as long
as it touches on the same topic as the gold answer, it should be counted as CORRECT.
For time related questions, the gold answer will be a specific date, month, year, etc. The
generated answer might be much longer or use relative time references (like "last Tuesday" or
"next month"), but you should be generous with your grading - as long as it refers to the same
date or time period as the gold answer, it should be counted as CORRECT. Even if the format
differs (e.g., "May 7th" vs "7 May"), consider it CORRECT if it's the same date.
Now it's time for the real question:
Question: {question}
Gold answer: {standard_answer}
Generated answer: {generated_answer}
First, provide a short (one sentence) explanation of your reasoning, then finish with CORRECT or
WRONG. Do NOT include both CORRECT and WRONG in your response, or it will break the evaluation
script.
Just return the label CORRECT or WRONG in a json format with the key as "label".
```

E.2 LongMemEval-S Benchmark

E.2.1 Question Answering Prompt

We follow the default non-CoT template from LongMemEval (Wu et al., 2025):

```
I will give you several related memories between you and a user. Please answer the question based
on the relevant memories.
Related Memories:
{memories}
Current Date: {current_date}
Question: {question}
Answer:
```

E.2.2 LLM-as-Judge Evaluation Prompt

LongMemEval uses task-specific evaluation prompts. For most tasks (SSU, SSA, MS):

```
I will give you a question, a correct answer, and a response from a model. Please answer yes if
the response contains the correct answer. Otherwise, answer no. If the response is equivalent to
the correct answer or contains all the intermediate steps to get the correct answer, you should
also answer yes. If the response only contains a subset of the information required by the
answer, answer no.
Question: {question}
Correct Answer: {answer}
Model Response: {response}
Is the model response correct? Answer yes or no only.
```

For temporal reasoning tasks, off-by-one tolerance is applied:

```
I will give you a question, a correct answer, and a response from a model. Please answer yes if
the response contains the correct answer. Otherwise, answer no. If the response is equivalent to
the correct answer or contains all the intermediate steps to get the correct answer, you should
also answer yes. If the response only contains a subset of the information required by the
answer, answer no. In addition, do not penalize off-by-one errors for the number of days. If the
question asks for the number of days/weeks/months, etc., and the model makes off-by-one errors
(e.g., predicting 19 days when the answer is 18), the model's response is still correct.
Question: {question}
Correct Answer: {answer}
Model Response: {response}
Is the model response correct? Answer yes or no only.
```

For knowledge update tasks:

```
I will give you a question, a correct answer, and a response from a model. Please answer yes if
the response contains the correct answer. Otherwise, answer no. If the response contains some
previous information along with an updated answer, the response should be considered as correct
as long as the updated answer is the required answer.
Question: {question}
Correct Answer: {answer}
Model Response: {response}
Is the model response correct? Answer yes or no only.
```

For single-session preference tasks:

```
I will give you a question, a rubric for desired personalized response, and a response from a
model. Please answer yes if the response satisfies the desired response. Otherwise, answer no.
The model does not need to reflect all the points in the rubric. The response is correct as long
as it recalls and utilizes the user's personal information correctly.
Question: {question}
Rubric: {rubric}
Model Response: {response}
Is the model response correct? Answer yes or no only.
```

E.3 TiMem System Prompts

We present key prompt templates used in TiMem's internal processing pipeline:

E.3.1 L1 Segment Memory Consolidator

```
You are a dialogue memory generator. Your task is to write a fragment memory that captures only
the NEW facts from the "Current Conversation" (do not repeat anything already covered in
"Historical Memories").
Core principle:
Convert dialogue from first-person to third-person narration, preserving as much substantive
information content from the original as possible, excluding only confirmed non-informative
words.
What to preserve:
- All substantive information: people, events, times, places, causes, results, numbers, specific
descriptions
- Original wording: Keep specific terms used in dialogue for titles, item names, activity
descriptions, etc, numbers use Arabic numerals
- Emotional expressions: Retain explicit emotions and attitudes from original (like "happy",
"worried", "likes"), but avoid adding subjective inferences not present in original
What to exclude:
Only exclude purely functional words: greetings ("hi""bye"), confirmation words
("uh-huh""okay""yes"), meaningless fillers ("um""you know""like")
Time normalization:
- Preserve the original relative time expressions exactly as written (e.g., "last night", "this
morning", "last Friday"). DO NOT convert relative time to absolute dates.
Style:
- Use English third-person narration.
- Write plain sentences (no lists/numbering/Markdown). Aim for 2-4 sentences, but allow longer to
retain essential details.
- Use exact proper nouns as in the dialogue; do not replace/expand/infer names, organizations, or
locations.
- Each memory should focus on one core fact or closely related fact group; avoid packing too many
unrelated details into a single entry.
Inputs:
- Historical memories (do not repeat): {previous_summary}
- Current conversation: {new_dialogue}
Please generate a fragment memory that contains ONLY the new facts. If the current conversation
has no substantial new content, provide a minimal 1-2 sentence summary of the core topic or
attitude expressed in this turn (do NOT output "no significant additions" or similar empty
statements).
```

E.3.2 Recall Gating Prompt for Simple Queries

```
Filter memories for simple fact query (Complexity 0).
Strategy: Aggressive filtering - Keep only direct answers
Target: 3-8 memories
## Filtering Rules
1. KEEP if memory directly answers the question
2. KEEP if memory provides essential context (time/location of the fact)
3. EXCLUDE if related but does not contribute to answer
4. EXCLUDE if different topic entirely
## Instructions
- Be strict: Only keep memories that help answer the specific question
- Remove noise: Exclude tangentially related memories
- Aim for 3-8 memories total
Question: {question}
Candidate memories ({total_count} total):
{numbered_memories}
Return IDs to keep (JSON format):
{{"relevant_ids": [1, 2, 3, ...]}}
```

E.3.3 Recall Planner Prompt

```
You are a professional query intent analysis expert. Please select the most appropriate retrieval
method based on the question type, and extract keywords.
Critical Judgment Principles:
- If the question requires understanding the user's preferences, habits, values, personality
traits, or historical behavior patterns to answer correctly, classify as "Deep Retrieval" (2)
- If the question involves reasoning, prediction, evaluation, subjective judgment, or
hypothetical scenarios, classify as "Deep Retrieval" (2)
- If the question requires integrating behaviors across multiple time points, multiple choices,
or long-term trends to answer, classify as "Deep Retrieval" (2)
- Only single factual queries (who, when, where, what specific action) should be classified as
"Simple Retrieval" (0)
Retrieval Type Definitions:
0 - Simple Retrieval (Factual Questions):
- Questions answerable by retrieving a single fact fragment
- Characteristics: Explicit time, location, person, event, or other objective fact queries
- Examples: "Where does X work?" "When did X go to location Y?" "Which meeting did X attend?"
- Key: Answer is an explicitly recorded fact, no reasoning or preference judgment needed
1 - Hybrid Retrieval (Multi-Fact Integration Questions):
- Questions requiring integration of multiple fact memories to answer
- Characteristics: Need to enumerate, summarize, or compare multiple facts, but no deep reasoning
required
- Examples: "What activities did X participate in?" "What topics did X and Y discuss?" "Where has
X been?"
- Key: Need to aggregate multiple facts, but still objective information integration
2 - Deep Retrieval (Personalized Reasoning Questions):
- Questions requiring reasoning based on user's deep personalized information (preferences,
habits, values, personality) to answer
- Core Characteristics:
  * Need to understand user's stable preferences (what they like/dislike, values, interests)
  * Need to infer user's future behavior or likely choices ("Would like...?" "Suitable for...?"
"Would choose...?")
  * Need to evaluate or judge user's personality traits, behavior patterns, cognitive style
  * Involves subjective judgment, evaluation, recommendation, prediction, hypothetical questions
  * Need to infer user's attitude or tendency based on historical behavior patterns
- Examples: "Would X enjoy a beach vacation?" "Is X an extroverted person?" "Might X be
interested in programming?" "Does X prioritize career or family more?"
- Key: Answer requires synthesizing user's deep traits and preferences, not directly recorded
facts
Judgment Process:
1. First identify: Does the question require user's preferences/habits/personality/values? If yes
→ Deep Retrieval (2)
2. Second identify: Does the question require reasoning/prediction/evaluation/subjective
judgment? If yes → Deep Retrieval (2)
3. Third identify: Does the question require summarizing multiple fact fragments? If yes → Hybrid
Retrieval (1)
4. Finally: If only single explicit fact needed → Simple Retrieval (0)
Keyword extraction requirements:
1. Extract 1-3 most important keywords from the question
2. Exclude common stopwords (such as: the, a, in, is, have, and, or, with, etc.)
3. STRICTLY FORBIDDEN: Never include any personal names, usernames, or names
4. FOCUS ONLY ON: Action words, object names, location types, concept words, adjectives and other
non-name key concepts
Question: {question}
Please carefully analyze the essential needs of the question and output in the following JSON
format:
{\n
  "complexity": 0/1/2,\n
  "keywords": ["keyword1", "keyword2", "keyword3"]\n
}
```