AdaMem: Adaptive User-Centric Memory for Long-Horizon Dialogue Agents

Anonymous ACL submission

Abstract

Long-horizon dialogue agents require memory systems that can preserve user-centric evidence, connect related experiences, and adapt retrieval to different questions. Existing memory approaches often rely on semantic similarity or static segmentation, which can miss personal attributes, weaken temporal coherence, and retrieve evidence at an unsuitable granularity. We propose **AdaMem**, an adaptive user-centric memory framework that organizes dialogue history into participant-specific working, episodic, persona, and graph memories. At inference time, AdaMem resolves the target participant, plans question-conditioned retrieval over semantic and relation-aware evidence, and uses role-specialized agents to synthesize evidence and generate concise answers. On LoCoMo and PERSONAMEM, AdaMem achieves state-of-the-art performance, showing that structured, adaptive, and user-centric memory improves long-horizon dialogue reasoning and user modeling. The code will be released upon acceptance.

1 Introduction

Recent advances in large language model (LLM)-based agents have enabled increasingly capable systems for open-ended dialogue, multi-step reasoning, and interactive assistance (Schlegel et al., 2025; Schmidgall et al., 2025). Yet these settings are inherently long-horizon: an agent must continually accumulate information across many turns, preserve salient details as user goals evolve, and recover the right evidence when it becomes relevant again. This makes memory a central requirement rather than a peripheral add-on. A useful memory system should not only store past interactions, but also organize them in a form that remains queryable, coherent, and robust as the conversation grows longer and more diverse. Otherwise, memory can become redundant, fragmented, or misaligned with the needs of downstream reasoning, leading to inconsistent behavior and poorly grounded responses.

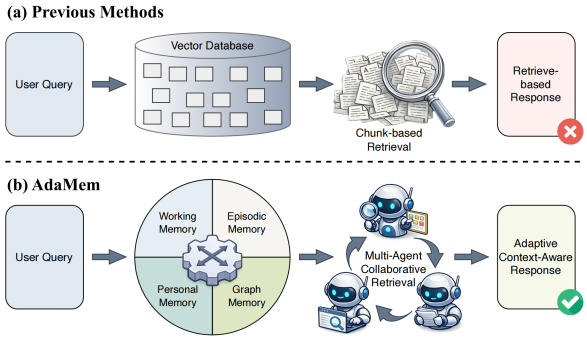

Figure 1: **Comparison of previous methods and AdaMem.** Conventional approaches rely on fixed-length chunks or coarse summaries with semantic retrieval, while AdaMem emphasizes user-centric adaptive structured memories and multi-agent collaborative retrieval.

Many recent agent frameworks therefore augment LLMs with explicit external memory modules that support incremental writing, updating, and retrieval throughout interaction (Yan et al., 2025; Rasmussen et al., 2025). Despite this progress, current approaches still face three limitations.

Limitation 1: Memory systems that rely primarily on semantic retrieval may overlook evidence that is not lexically or semantically similar to the query, but is still crucial for user-centric understanding, such as stable preferences, personal attributes, or broader behavioral patterns.

Limitation 2: When related experiences are stored as isolated fragments, their temporal and causal coherence can be weakened, making it difficult to reconstruct how events unfolded and how different pieces of evidence should be connected during reasoning.

Limitation 3: Different questions require different memory structures and retrieval strategies.

Table 1: **Comparison of long-horizon dialogue memory capabilities.**

Method	Memory-Reasoning Synergy	Role-Specialized Agents	Relation-Aware Graph	Training-Free
MIRIX	✗	✓	✗	✓
LightMem	✗	✗	✗	✓
MEM1	✓	✗	✗	✗
AdaMem (Ours)	✓	✓	✓	✓

As illustrated in Figure 1, many systems construct memory entries using either fixed-length text chunks (Zhang et al., 2025b; Wu et al., 2025). Such static segmentation is often a poor fit for long-horizon reasoning: overly coarse memories may introduce substantial irrelevant context, while overly fine-grained fragments can obscure dependencies across events and topics.

These limitations suggest that effective long-horizon memory should be both *structured* and *adaptive*: it should preserve information at multiple levels of abstraction while dynamically selecting retrieval routes that match each question. Table 1 compares recent memory systems (Wang and Chen, 2025; Fang et al., 2026; Zhou et al., 2026) along these long-horizon dialogue capabilities.

Motivated by this observation, we propose *AdaMem*, an adaptive user-centric memory framework for long-horizon dialogue agents. AdaMem can be viewed as a unified memory interface that jointly determines *who* the query refers to, *what* memory granularity should be searched, *whether* structural expansion is needed, and *how* heterogeneous evidence should be fused. Concretely, AdaMem maintains participant-specific working, episodic, persona, and graph memories, organizing evidence around the user and the assistant in a unified yet target-aware manner. Unlike prior systems that rely on a monolithic controller for memory writing, retrieval, verification, and response generation, AdaMem uses a role-specialized agent pipeline. The Memory Agent maintains structured memories, the Research Agent performs question-conditioned retrieval and evidence integration with reflection, and the Working Agent turns the collected evidence into a concise answer. This decomposition reduces interference between memory maintenance and answer-time reasoning, enabling finer control over retrieval, verification, and memory evolution. Together, these design choices recover user information beyond pure semantic similarity, preserve links across related events, and adapt retrieval granularity to the demands of the question.

In summary, our contributions are:

- We introduce AdaMem, an adaptive user-centric memory framework for long-horizon dialogue agents that organizes dialogue history into complementary working, episodic, persona, and graph-based memory structures.
- We propose a question-conditioned retrieval and response pipeline that resolves target participants, invokes relation-aware graph expansion only when needed, and uses specialized agents for evidence synthesis and answer generation.
- We validate our approach on the LoCoMo and PERSONAMEM benchmarks, achieving state-of-the-art performance and demonstrating its strong effectiveness.

2 Related Works

2.1 Agentic Memory

Recent research on memory systems for large language model agents has evolved from simple context extension toward more structured and adaptive management. Early approaches typically process long contexts by partitioning them into smaller chunks (Zhong et al., 2024; Mei et al., 2025; Wang et al., 2025a; Liu et al., 2024). Subsequent work introduces more advanced memory mechanisms. For instance, MemGPT (Packer et al., 2023) manages long-term memory through paging and segmentation. Later studies explore more modular and system-level designs; Mem0 (Chhikara et al., 2025) abstracts memory as an independent layer for long-term management. Meanwhile, several works investigate structured memory representations to improve organization, including graph-based memories such as A-Mem (Xu et al., 2025) and semantic memory structures built upon events or temporal knowledge graphs, such as Zep (Rasmussen et al., 2025). Recent systems such as LightMem (Fang et al., 2026) and MEM1 (Zhou et al., 2026) further explore efficient memory-augmented generation and memory–reasoning synergy, respectively. LightMem emphasizes lightweight memory compression and efficient retrieval, while MEM1 studies how a memory module can be trained to synergize with long-horizon reasoning. AdaMem addresses a different but complementary axis: explicit user-centric organization at inference time. It maintains functionally differentiated memories,

preserves participant ownership, and selects retrieval routes according to the target participant and question intent. Despite recent progress, many existing approaches still rely on static retrieval strategies or largely unstructured storage, which may lead to information fragmentation and limited coordination across different abstraction levels and task categories. Designing a unified and adaptive memory system that robustly supports long-term interactions therefore remains an important challenge.

2.2 Multi-Agent LLMs

Recent studies have shown that multi-agent LLMs can effectively address complex tasks by enabling role specialization, collaborative problem solving, and interactive decision making (Tran et al., 2025; Cemri et al., 2025; Hammond et al., 2025; Yue et al., 2025; Zhang et al., 2025a). At the same time, an increasing number of works on agentic memory aim to enhance the capability of large language model agents to model and retain long-term information (Yan et al., 2025; Wang et al., 2025b). Although these efforts provide useful perspectives on coordination and task decomposition, they typically do not focus on how long-horizon dialogue evidence should be organized and adaptively retrieved in a user-centric manner. A recent study, MIRIX (Wang and Chen, 2025), takes an initial step in this direction by introducing specialized agents for memory organization. AdaMem differs from MIRIX by making the retrieval interface itself adaptive: role specialization is coupled with participant-aware multi-view memory, question-conditioned route planning, relation-aware expansion, and unified evidence fusion. Thus, the key distinction is not the use of multiple agents alone, but the tight coupling between agent roles and the structured memory operations they invoke. Motivated by these observations, our work draws on role specialization from the multi-agent literature, but focuses primarily on user-centric memory construction and question-conditioned retrieval for long-horizon dialogue agents.

3 Approach

We present AdaMem, a memory-augmented dialogue framework that continuously organizes user-centric memories at inference time and answers questions through adaptive retrieval over heterogeneous memory sources. AdaMem consists of four tightly coupled components: memory construction, question-conditioned retrieval planning, evidence fusion, and response generation. The overall pipeline is illustrated in Figure 2.

3.1 Method Overview

Given a dialogue history $\mathcal{D} = \{u_t\}_{t=1}^{T}$ and a question q, AdaMem returns an answer a grounded in dynamic memories. AdaMem maintains participant-specific memory bundles for the user and the assistant, and each bundle contains four memory structures:

- **Working Memory** $\mathcal{M}^{\mathrm{w}}$: a bounded FIFO buffer that preserves recent conversational context and short-term discourse states.
- **Episodic Memory** $\mathcal{M}^{\mathrm{e}}$: long-term structured records including events, facts, attributes, and topic-centric summaries.
- **Persona Memory** $\mathcal{M}^{\mathrm{p}}$: compact user profiles distilled from episodic evidence to capture relatively stable preferences and traits.
- **Graph Memory** $\mathcal{G}$: a heterogeneous graph connecting messages, topics, facts, attributes, and event or persona snapshots for relation-aware retrieval. Detailed graph construction rules are provided in Appendix C.

This participant-specific organization is important because many questions in multi-party conversations implicitly target one speaker, both speakers, or an ambiguous referent. Accordingly, AdaMem follows a single end-to-end pipeline: it first writes each utterance into structured participant memories, then resolves the likely target participant for q, retrieves evidence from one or more memory channels, and finally produces the answer from the fused evidence set $\mathcal{R}(q)$.

3.2 Memory Construction

Message understanding and normalized write. For each incoming utterance u_t from participant p, the Memory Agent first produces a normalized record z_t containing a short summary, topic, attitude, reason, factual snippets, attributes, timestamp, and speaker identity. All downstream memory updates operate on z_t rather than on raw free-form text, so the same canonical parse is reused across working, episodic, persona, and graph memories. This design reduces prompt drift across memory modules and makes the write path explicit.

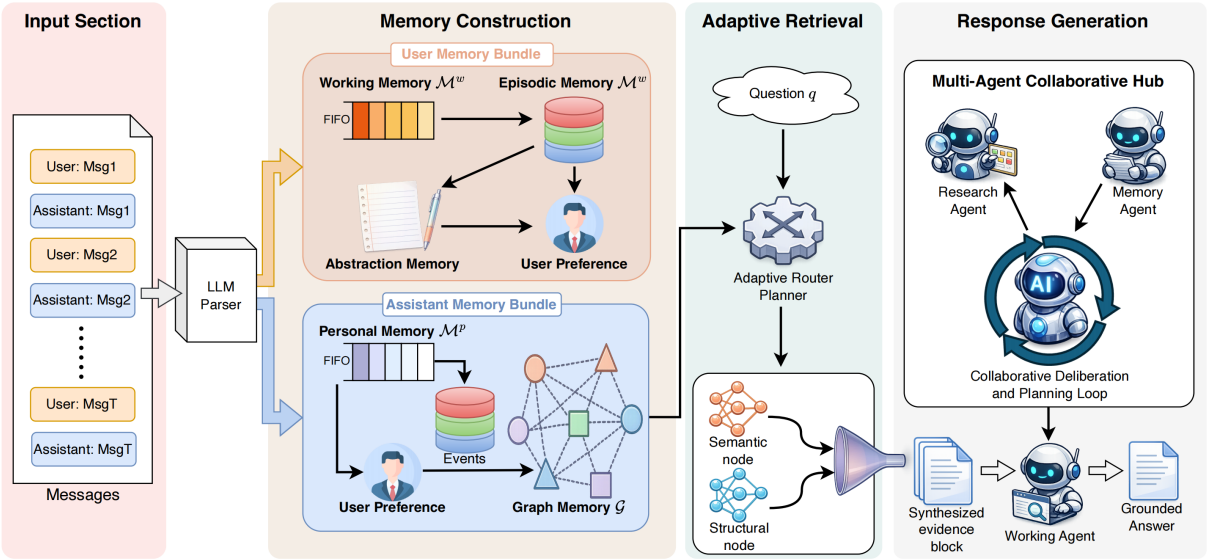

Figure 2: **Model overview.** Dialogue history is organized into working, episodic, persona, and graph memories, and question answering is performed through target-aware, question-conditioned retrieval and role-specialized evidence synthesis.

Working-to-episodic consolidation. For each participant, working memory is a bounded FIFO queue with capacity C_{w}. When the queue becomes full, AdaMem pops the oldest contiguous segment of r messages and consolidates that segment into $\mathcal{M}^{\mathrm{e}}$. The segment is therefore determined by recency order rather than by question-aware salience, preventing future questions from implicitly influencing write-time memory selection. Within the popped segment, three router modules independently process event-, fact-, and attribute-level evidence. Each router predicts one of `ADD`, `UPDATE`, or `IGNORE`, together with a target key if an existing entry should be revised. These three operators serve as minimal primitives for incremental memory maintenance: `ADD` introduces genuinely new information, `UPDATE` refines or corrects an existing entry when new evidence changes it, and `IGNORE` suppresses redundant or semantically equivalent content. This design controls memory growth while preserving provenance, because revised memories remain linked to their supporting dialogue evidence rather than being destructively overwritten. The resulting updates populate event, fact, and attribute stores, while the original messages remain available as cacheable provenance for later evidence recovery.

Topic regrouping, persona refresh, and graph synchronization. After consolidation, AdaMem converts fine-grained episodic keys into reusable higher-level memories in two stages. First, event or attribute keys are embedded and linked by a sparse nearest-neighbor graph: each key is connected to its most similar peer, and the connected components define merge groups. Second, an LLM merge prompt rewrites each group into a topic-centric or aspect-centric summary. Topic groups are used to build topic episodic memories and preference-oriented persona descriptors, while clustered attributes are merged into aspect-based persona summaries. In parallel, both message-level and consolidated records are indexed into $\mathcal{G}$, so later retrieval can move between local discourse evidence and longer-term structured abstractions.

3.3 Question-Conditioned Retrieval

Target participant resolution. Before retrieval, AdaMem resolves whether q refers to the user, the assistant, both, or an ambiguous participant. The implementation uses a lightweight four-way resolver based on explicit participant mentions; if the referent is ambiguous, the system avoids a hard commitment and instead runs a small retrieval on both participant bundles before fusion. This makes target uncertainty visible to the retrieval stage rather than hiding it inside later answer generation.

Route planning. Given q, AdaMem builds a question-conditioned route plan $\pi(q)$ that specifies whether graph expansion should be used, how far it may propagate, how many graph seeds are activated, which edge-type priors are applied, and

how baseline and graph evidence will be fused. The planner first applies deterministic cue detection over temporal, relational, attribute, and single-hop question patterns. These cues initialize a rule-based plan. We use transparent heuristics rather than a learned routing controller because the benchmarks do not provide dense supervision for retrieval-policy learning. To keep the route planner reproducible and avoid benchmark-specific tuning, all cue rules and default ranges are fixed before evaluation. When the question remains uncertain, an optional LLM refinement step can revise the plan, but only within narrow bounded ranges so that the planner remains conservative and semantic retrieval stays dominant for simple questions. As a result, single-hop factoid questions usually remain on lightweight semantic retrieval, whereas temporal or causal questions trigger broader structural exploration.

Target-aware baseline retrieval. For a selected participant bundle, baseline retrieval aggregates semantic candidates from persona summaries, episodic facts, and topic-linked messages:

$$\mathcal{R}_{\text{base}}(q) = \text{TopK}(\mathcal{R}_{\text{attr}}(q) \cup \mathcal{R}_{\text{fact}}(q) \cup \mathcal{R}_{\text{topic}}(q)), \tag{1}$$

where attribute candidates come from persona summaries, fact candidates come from episodic fact memory, and topic candidates are linked back to original messages through topic-to-message maps. Beyond pure top-K aggregation, AdaMem applies two recovery mechanisms. First, high-confidence fact hits reactivate their supporting discourse messages. Instead of attaching all neighboring contexts, AdaMem uses a score-drop rule over ranked fact matches to decide how many supporting messages should be reintroduced. Second, a lightweight keyword backoff over a word-to-detail index recalls raw messages that are weakly covered by dense retrieval but lexically salient for the question.

Graph retrieval and evidence fusion. When $\pi(q)$ requests relation-aware evidence, AdaMem selects top semantic seed nodes in $\mathcal{G}$ and performs bounded multi-hop expansion. If a seed node u reaches a neighbor v through edge type e, the propagated score is updated by a fixed multiplicative rule

$$s_{d+1}(v) = s_d(u) \cdot w_e \cdot \lambda, \tag{2}$$

where w_e is the edge-type prior and λ is a hop-decay factor. These priors act as lightweight retrieval biases rather than learned graph-reasoning parameters, and they are fixed before evaluation unless the bounded route planner applies a clipped question-type adjustment. Owner-aware filtering is applied only when the target participant is unambiguous. This yields $\mathcal{R}_{\text{graph}}(q)$. For a candidate memory item m and query q, AdaMem then assigns a final retrieval score by fusing baseline and graph evidence:

$$\begin{aligned}\text{score}(m \mid q) = \alpha\, s_{\text{base}}(m, q) + \beta\, s_{\text{graph}}(m, q) \\ +\gamma\, s_{\text{recency}}(m) + \delta\, s_{\text{fact}}(m).\end{aligned} \tag{3}$$

Here $s_{\text{base}}(m, q)$ and $s_{\text{graph}}(m, q)$ are rank-derived scores from baseline and graph retrieval, respectively; $s_{\text{recency}}(m)$ is a linear recency prior over the merged candidate list; and $s_{\text{fact}}(m)$ is a small confidence bonus for items also supported by fact retrieval. The coefficients $\alpha, \beta, \gamma, \delta$ control the relative influence of these four signals. A global default prior is fixed before evaluation, and optional question-conditioned adjustments are clipped to small ranges, making fusion interpretable rather than a learned task-specific reranker.

3.4 Multi-Agent Collaboration

Memory Agent. The Memory Agent is responsible for online message understanding and memory updates. For each new utterance, it extracts a normalized representation, writes it into the participant-specific working memory, triggers working-to-episodic consolidation when the short-term buffer saturates, and synchronizes the resulting message and memory artifacts to graph memory. Persona descriptors are then refreshed from aggregated episodic evidence at the indexing stage, so the system maintains both up-to-date local context and compact long-term user models.

Research Agent. At question answering time, the Research Agent performs iterative evidence gathering over the unified retrieval interface described above. It follows a *Planning* → *Search* → *Integrate* → *Reflection* loop: it first decomposes the information needed to answer q, then issues one or more retrieval requests through the same participant-aware retrieval API, integrates newly recovered evidence into a consolidated research summary, and decides whether additional search is still necessary. Importantly, this agent-level planning is distinct from the route plan $\pi(q)$: the Research Agent decides *what* missing information to

ask for, while the route planner described above decides *how* each retrieval call is executed.

Working Agent. The Working Agent converts the research summary into the final concise answer. It conditions generation primarily on the integrated summary returned by the Research Agent and, when needed, supplements it with high-confidence persona attributes or factual snippets as auxiliary grounding. This separation allows evidence collection and answer realization to be optimized for different roles while preserving a single memory interface.

AdaMem answers each question through a fixed collaboration order among these roles. As a result, response generation remains tightly coupled with the same user-centric memory interface and retrieval backbone introduced above, while allocating explicit deliberation to multi-step evidence synthesis before final answer generation.

4 Experiments

4.1 Experimental Setting

Benchmarks. We evaluate AdaMem on two long-horizon dialogue benchmarks. LoCoMo (Maharana et al., 2024) contains dialogue histories averaging 35 sessions and 9,000 tokens, and evaluates single-hop, multi-hop, temporal, and open-domain question answering. PERSONAMEM (Jiang et al., 2025) tests whether models can maintain evolving user representations and generate personalized responses across multi-session interactions, with results reported over seven user-modeling categories.

Evaluation Metrics. For the LoCoMo benchmark, we follow the standard evaluation protocol and report F1 and BLEU-1 as the primary metrics. For the PERSONAMEM benchmark, which primarily consists of multiple-choice questions, we evaluate model performance using accuracy.

4.2 Implementation Details

We conduct experiments on both closed-source APIs and open-source models. The closed-source models include GPT-4.1-mini and GPT-4o-mini (Achiam et al., 2023). For open-source models, we evaluate Qwen3-4B-Instruct and Qwen3-30B-A3B-Instruct (Yang et al., 2025). Experiments are conducted on a NVIDIA RTX A800 GPU. We ensure that the AdaMem framework uses the same backbone model as the response generator. To ensure reproducibility, the temperature is fixed to 0 in all experiments. For the RAG component in our framework, the retrieval top-k is set to 10, and the maximum number of retrieval iterations L_i is limited to 2 for efficiency. All memory embeddings are computed using the `all-MiniLM-L6-v2` model (Reimers and Gurevych, 2019). More implementation details are provided in Appendix B and prompt templates are provided in Appendix E.

4.3 Comparison to competitive approaches

Baselines. We compare AdaMem with five representative open-source memory frameworks: (1) *MemGPT* (Packer et al., 2023), a framework that addresses long-context limitations through an OS-inspired memory management mechanism; (2) *A-Mem* (Xu et al., 2025), an agentic memory system that enables dynamic organization and evolution of long-term memory; (3) *Mem0* (Chhikara et al., 2025), a memory-centric architecture that provides scalable long-term memory; (4) *LangMem* (Chase, 2024), a framework that explicitly models long-term memory; (5) *Zep* (Rasmussen et al., 2025), a memory layer that represents conversational memory using a temporally aware knowledge graph. We did not include MIRIX (Wang and Chen, 2025), LightMem (Fang et al., 2026), and MEM1 (Zhou et al., 2026) as baselines in this work because their open-source releases were incomplete for evaluation on the LoCoMo and PERSONA benchmarks during our experimental phase.

Results on LoCoMo. Table 2 summarizes LoCoMo results with closed-source backbones. AdaMem achieves the best overall F1 with both GPT-4.1-mini (44.65, +4.4% relative improvement) and GPT-4o-mini (41.84, +12.8%), with particularly large gains on temporal questions. Although AdaMem is not strongest on every open-domain category, its overall results show robust gains across backbones. Open-source model results are reported in Table 4, and case studies are provided in Appendix D.

Results on PERSONAMEM. To evaluate the generalization capability of AdaMem across benchmarks, we further evaluate it on the PERSONAMEM benchmark. As shown in Table 3, AdaMem achieves 63.25% accuracy, outperforming all baselines with a relative improvement of 5.9%. Notably, AdaMem demonstrates a substantial advantage on the generalize to new scenarios task, achieving a relative improvement of 27.3%. These consistent gains across benchmarks with dif-

Table 2: **Performance on the LoCoMo benchmark.** The best performance is highlighted in **bold**, and the second-best is underlined.

Model	Method	Multi-hop		Temporal		Open Domain		Single-hop		Overall	
		F1	BLEU-1	F1	BLEU-1	F1	BLEU-1	F1	BLEU-1	F1	BLEU-1
GPT-4.1-mini	MemGPT	10.35	10.26	30.06	24.03	28.35	23.49	22.95	17.10	22.50	17.64
	A-Mem	11.16	11.07	42.30	33.75	**30.69**	**25.38**	24.84	18.54	26.37	20.70
	Mem0	35.19	26.55	34.38	29.07	20.88	15.57	42.66	36.90	38.16	32.04
	LangMem	36.45	28.53	42.57	35.91	28.80	23.13	44.82	38.25	41.76	35.10
	Zep	26.73	17.91	20.97	17.55	21.24	16.92	39.96	35.10	32.40	27.09
	AdaMem	**37.70**	**32.08**	**55.90**	**42.37**	25.87	21.70	**44.84**	**40.02**	**44.65**	**37.92**
GPT-4o-mini	MemGPT	10.08	9.99	29.34	23.49	27.72	22.95	22.41	16.74	21.96	17.28
	A-Mem	10.89	10.80	41.31	33.03	**29.97**	**24.84**	24.30	18.09	25.74	22.95
	Mem0	30.60	22.50	39.60	33.57	24.21	17.28	39.69	30.60	37.08	30.51
	LangMem	30.24	21.60	28.89	23.67	26.55	21.24	35.01	29.88	32.31	26.55
	Zep	24.57	17.28	39.96	34.02	20.43	14.04	35.46	30.06	33.48	27.63
	AdaMem	**35.18**	**26.32**	**51.49**	**37.06**	25.82	21.46	**42.21**	**36.43**	**41.84**	**33.78**

ferent data distributions suggest that AdaMem generalizes effectively to diverse long-term reasoning scenarios.

4.4 Ablation Study

Unless otherwise stated, we perform all ablations of AdaMem on the LoCoMo benchmark using the GPT-4.1-mini backbone.

Model Sizes. We further study whether AdaMem remains effective when the response backbone changes in both scale and model family. Table 4 compares two closed-source GPT-4 backbones with two open-source Qwen3 models while keeping the AdaMem framework unchanged. AdaMem remains effective across backbone families: Qwen3-4B-Instruct reaches 36.78 F1, and Qwen3-30B-A3B-Instruct improves to 38.58 F1. The largest gains from scaling appear on open-domain and temporal questions, suggesting that stronger backbones better exploit structured evidence without changing the memory framework.

Components. Table 5 shows that each main component contributes to AdaMem. Removing graph memory causes the largest drop, from 44.65 to 42.63 F1, confirming the importance of relation-aware evidence recovery. Removing fusion or multi-agent collaboration also degrades performance, indicating that structured retrieval, evidence aggregation, and role-specialized reasoning provide complementary gains.

The component results also indicate how AdaMem improves beyond end-task gains. Removing graph memory mainly weakens the ability to recover temporally linked or cross-turn evidence that is not retrieved by semantic similarity alone. Removing fusion hurts because baseline retrieval, graph expansion, recency, and factual support provide complementary but noisy signals; using a single source makes the final evidence set less stable. Replacing the role-specialized pipeline with a single agent causes a smaller but consistent drop, suggesting that most gains come from better evidence coverage, while multi-agent collaboration further improves how the retrieved evidence is organized before answer generation. The case studies in Appendix D illustrate the same pattern: AdaMem succeeds when participant-aware structured evidence disambiguates a lexically broad query, and still fails when implicit entity linking or relative-time normalization is absent from memory. These observations suggest that the framework's gains come from coordinated evidence recovery rather than from any single dominant module.

Hyperparameters. We study the sensitivity of AdaMem to the retrieval top-K and the maximum number of Research Agent iterations L_i in Figure 3. With $L_i = 2$, performance improves from $K = 5$ to $K = 10$ but gains little at $K = 15$, suggesting that larger candidate pools add noise after the main evidence is covered. With $K = 10$, both F1 and BLEU-1 peak at $L_i = 2$: one iteration under-explores multi-step questions, while three iterations can introduce redundant evidence. We therefore use $K = 10$ and $L_i = 2$ as a stable performance–efficiency trade-off.

4.5 Efficiency Analysis

We analyze the performance-efficiency trade-off of AdaMem. Table 6 reports the average input token budget, end-to-end inference latency, and overall

Table 3: **Performance on the PERSONAMEM benchmark.**

Method	Recall user shared facts	Suggest new ideas	Track full preference evolution	Revisit reasons behind preference updates	Provide preference-aligned recommendations	Generalize to new scenarios	**Average**
A-Mem	63.01	**27.96**	54.68	85.86	**69.09**	57.89	59.75
Mem0	32.13	15.05	54.68	80.81	52.73	57.89	48.55
LangMem	31.29	24.73	53.24	81.82	40.00	8.77	40.64
AdaMem	**67.81**	21.51	**61.15**	**89.90**	65.45	**73.68**	**63.25**

Table 4: **Ablation on model sizes.**

Model	**Multi-hop**		**Temporal**		**Open Domain**		**Single-hop**		**Overall**	
	F1	**BLEU-1**	**F1**	**BLEU-1**	**F1**	**BLEU-1**	**F1**	**BLEU-1**	**F1**	**BLEU-1**
GPT-4o-mini	35.18	26.32	51.49	37.06	25.82	21.46	42.21	36.43	41.84	33.78
GPT-4.1-mini	37.70	32.08	55.90	42.37	25.87	21.70	44.84	40.02	44.65	37.92
Qwen3-4B-Instruct	29.50	23.35	40.45	29.74	20.05	17.68	39.74	34.98	36.78	30.68
Qwen3-30B-A3B-Instruct	30.82	25.17	43.51	30.90	26.55	23.31	40.68	35.99	38.58	32.16

Table 5: **Ablation on key components.**

Configuration	**Graph**	**Fusion**	**Multi-agent**	**F1**	**BLEU-1**
AdaMem (full)	✓	✓	✓	**44.65**	**37.92**
w/o graph	✗	✓	✓	42.63	35.85
w/o fusion	✓	✗	✓	42.77	36.26
w/o multi-agent	✓	✓	✗	43.24	36.34

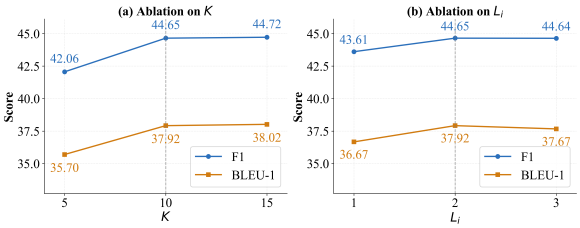

Figure 3: **Ablation on key hyperparameters.**

Table 6: **Performance-efficiency trade-off.**

Method	**F1**	**BLEU-1**	**Tokens**	**Latency (s)**
A-Mem	26.37	20.70	2720	3.227
Mem0	38.16	32.04	1340	3.739
Zep	32.40	27.09	2461	3.255
AdaMem	44.65	37.92	2248	4.722

F1 and BLEU-1. AdaMem is not the cheapest system in raw computation: Mem0 uses fewer input tokens and lower latency. However, AdaMem delivers the strongest answer quality, reaching 44.65 F1, which corresponds to absolute gains of +7.57 F1 over Mem0. Compared with A-Mem and Zep, AdaMem also achieves substantially better accuracy while operating with a token budget of the same order.

These results suggest that AdaMem's advantage does not come from aggressively minimizing retrieval cost, but from allocating a moderate amount of additional computation to recover higher-quality evidence. The extra latency is consistent with our design: question-conditioned route planning, graph-based expansion, and the role-specialized response loop introduce overhead, but they also improve evidence coverage and synthesis for long-horizon questions. Overall, AdaMem occupies a favorable operating point among memory-based methods, converting a moderate increase in token usage and latency into clearly stronger reasoning performance.

5 Conclusion

In this paper, we presented AdaMem, an adaptive user-centric memory framework for long-horizon dialogue agents. AdaMem combines participant-specific working, episodic, persona, and graph-based memories with question-conditioned retrieval planning and unified evidence fusion, enabling the system to retrieve and integrate evidence in a more structured and target-aware manner. Experiments on long-horizon reasoning and user modeling benchmarks demonstrate the effectiveness of our framework. In particular, AdaMem achieves state-of-the-art performance on LoCoMo, showing the value of adaptive memory organization and retrieval for complex multi-session interactions. More broadly, our results suggest that memory for long-horizon dialogue agents should move beyond uniform storage and fixed retrieval heuristics toward more adaptive, question-aware, and user-centric designs.

Limitations

AdaMem improves answer quality through structured memories, adaptive retrieval, and role-specialized evidence synthesis, but this design introduces additional system complexity, token cost, and latency. Like other persistent memory systems, it also depends on the quality of upstream parsing and backbone reasoning, so mistakes in target resolution, entity linking, or temporal normalization may affect later retrieval if they are consolidated into long-term memory. Finally, user-centric memory should be deployed with appropriate privacy controls, since it may retain user attributes, preferences, and interaction histories over long horizons.

Ethics Statement

AdaMem operates exclusively on publicly available benchmarks with open licenses, and does not involve human subjects, private data, or dual-use applications.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, and 1 others. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.

Harrison Chase. 2024. Langmem. https://github.com/langchain-ai/langmem. LangChain project. Accessed: 2025-07-20.

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

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. 2026. Lightmem: Lightweight and efficient memory-augmented generation. In *The Fourteenth International Conference on Learning Representations*.

Lewis Hammond, Alan Chan, Jesse Clifton, Jason Hoelscher-Obermaier, Akbir Khan, Euan McLean, Chandler Smith, Wolfram Barfuss, Jakob Foerster, Tomáš Gavenčiak, and 1 others. 2025. Multi-agent risks from advanced ai. *arXiv preprint arXiv:2502.14143*.

Bowen Jiang, Zhuoqun Hao, Young Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo Jose Taylor, and Dan Roth. 2025. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. In *Second Conference on Language Modeling*.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Liangwei Yang, Zuxin Liu, Juntao Tan, Prafulla K Choubey, Tian Lan, Jason Wu, Huan Wang, and 1 others. 2024. Agentlite: A lightweight library for building and advancing task-oriented llm agent system. *arXiv preprint arXiv:2402.15538*.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870.

Kai Mei, Xi Zhu, Wujiang Xu, Mingyu Jin, Wenyue Hua, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. 2025. Aios: Llm agent operating system. In *Second Conference on Language Modeling*.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*.

Katja Schlegel, Nils R Sommer, and Marcello Mortillaro. 2025. Large language models are proficient in solving and creating emotional intelligence tests. *Communications Psychology*.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Zhenhe Wu, ShuangZhi Wu, Zejun Ma, and Zhoujun Li. 2025a. Scm: Enhancing large language model

with self-controlled memory framework. In *International Conference on Database Systems for Advanced Applications*. Springer.

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

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. 2025b. Mem-{\alpha}: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*.

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 arXiv:2504.15965*.

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

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, and 1 others. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. Masrouter: Learning to route llms for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and 1 others. 2025a. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. In *Forty-second International Conference on Machine Learning*.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025b. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. 2026. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *The Fourteenth International Conference on Learning Representations*.

A Organization

The appendix contains the following sections:

- More implementation details are provided in Section B.
- Graph construction details are provided in Section C.
- Case studies are provided in Section D.
- Prompt templates are provided in Section E.

B More Implementation Details

This section documents the concrete decision rules used in our implementation so that AdaMem's gains can be attributed to its retrieval design rather than to hidden prompt engineering or benchmark-specific tuning. Unless otherwise stated, the benchmark settings below are fixed before evaluation and reused across runs. The repository also contains looser demo defaults for interactive examples; the paper numbers should therefore be interpreted with the benchmark configuration rather than the demo configuration.

B.1 Target Participant Resolution

AdaMem uses a lightweight four-way target resolver. If a question explicitly mentions the user name, the target is the user; if it explicitly mentions the assistant name, the target is the assistant; if it mentions both names, the target is `both`; otherwise the target is `ambiguous`. For `both` and `ambiguous`, the system does not force an early commitment. Instead, it performs a small retrieval on both participant bundles and merges the candidates before final reranking. In graph retrieval, owner-aware filtering is applied only when the target is unambiguous. We do not use a separately trained target-participant classifier.

B.2 Route Planning and Graph Expansion

The route planner first applies deterministic cue detection:

- temporal cues: "when", "date", "year", "month", "time", "last", "ago", "before", "after";
- relation cues: "why", "because", "cause", "how", "relationship", "connect", "between";
- attribute cues: "prefer", "like", "favorite", "personality", "trait", "attribute";
- single-hop cues: "who", "what", "where", "which", "name", "did", "does", "is", "was".

Temporal or relation cues enable graph retrieval, while single-hop questions without temporal or relation cues are kept on semantic retrieval whenever possible. In the hybrid setting, the rule-based plan is refined by an LLM only when the rule confidence is below a fixed threshold of 0.75. The refiner can update `use_graph`, graph seed count, hop depth, and fusion weights, but its outputs are clipped to narrow intervals to keep the planner conservative and to avoid unconstrained benchmark-specific adaptation.

B.3 Working Memory, Consolidation, and Context Reactivation

Each participant maintains a FIFO working-memory queue with capacity $C_{\mathrm{w}} = 20$. When the queue reaches capacity, AdaMem pops the oldest contiguous segment of $r = 5$ messages and sends that segment to episodic-memory consolidation. This segment is selected purely by recency order; no query-aware filtering is used at write time. During retrieval, fact-based context reactivation uses a score-drop heuristic rather than a fixed context window: ranked fact matches are scanned in order, and supporting context is reactivated until the first adjacent score drop exceeds $\tau_{\mathrm{drop}} = 0.1$.

B.4 Topic Clustering and Persona Refresh

We compute memory embeddings with `all-MiniLM-L6-v2` (Reimers and Gurevych, 2019). For both event/topic keys and attribute keys, AdaMem computes pairwise similarities, keeps only each key's most similar neighbor, and takes the connected components of this sparse similarity graph as merge clusters. Thus, clustering is sparse and threshold-free rather than based on a manually tuned similarity cutoff. Each event cluster is then passed to a topic-merge prompt that produces grouped topic memories, and the resulting grouped topics are further summarized into preference-oriented persona descriptors. Attribute clusters are merged into aspect-based persona summaries by a separate prompt.

B.5 Graph Propagation Priors

At indexing time, different edges receive fixed write-time strengths: message–topic `mentions` uses 0.75, message–fact `supports` uses 0.85, fact–event `supports` uses 0.80, message–attribute

Table 7: Default retrieval-time edge priors.

Edge type	Weight
`mentions`	0.75
`supports`	0.90
`same_topic`	0.55
`temporal_next`	0.70
`speaker_related`	0.60

Table 8: Default fusion prior in the benchmark configuration.

Weight	Value
α semantic rank	0.7
β graph rank	0.1
γ recency	0.1
δ fact confidence	0.1

`supports` uses 0.80, same-speaker temporal edges use 0.90 forward and 0.45 backward, global temporal edges use 0.70 forward and 0.35 backward, and `speaker_related` uses 0.65. At retrieval time, graph propagation uses the priors in Table 7. The planner can modestly increase the temporal prior for temporal questions and the speaker-related prior for attribute-heavy questions. Multi-hop propagation also uses a fixed hop-decay factor $\lambda = 0.85$.

B.6 Fusion Score Components and Weights

The four fusion terms in Eq. (3) are implemented as simple rank- and heuristic-based priors rather than learned scorers:

$$s_{\text{base}}(m, q) = \frac{1}{1 + \text{rank}_{\text{base}}(m)}, \quad s_{\text{graph}}(m, q) = \frac{1}{1 + \text{rank}_{\text{graph}}(m)}. \tag{4}$$

The recency term decreases linearly over the merged candidate list, and the factual term assigns a high bonus to messages already supported by baseline fact retrieval and a small residual bonus otherwise. For the benchmark configuration used in the released evaluation script, the default fusion prior is shown in Table 8. These values are fixed before evaluation rather than fit from supervision. The optional route refiner may adjust them per question, but only within clipped ranges around the default prior: $\alpha \in [0.65, 0.75]$, $\beta \in [0.05, 0.15]$, $\gamma \in [0.05, 0.15]$, and $\delta \in [0.05, 0.15]$.

Table 9: Representative released benchmark defaults.

Setting	Value
Working-memory capacity C_{w}	20
Consolidation segment length r	5
Baseline retrieval top-K	10
Fact reactivation threshold τ_{drop}	0.1
Base graph hop depth	1
Base graph seed count	2

B.7 Representative Benchmark Defaults

Table 9 summarizes the main released retrieval defaults that are hidden behind the implementation in the public code. The number of Research Agent iterations is configured separately as an experimental variable and is therefore not treated as an immutable retrieval default here.

C Graph Construction Details

This appendix section clarifies how AdaMem builds and retrieves from graph memory. Importantly, the graph is not produced by a standalone LLM graph-extraction prompt. Instead, each utterance is first converted into a normalized record containing topic, facts, attributes, summary, and auxiliary fields; graph indexing then applies deterministic node typing and edge-construction rules on top of this record.

C.1 Node and Edge Rules

For each processed utterance, AdaMem creates a `message` node storing the raw text, speaker, timestamp, and turn index. The normalized fields create additional typed nodes: `topic` nodes for message topics, `fact` nodes for extracted factual snippets, `attribute` nodes for user- or counterpart-related attributes, and `event` nodes for event abstractions linked from facts or persona snapshots. We then add a small set of typed edges:

- `mentions`: connects a message to its topic node;
- `supports`: connects a message to extracted fact or attribute nodes and connects fact nodes to their associated event nodes;
- `same_topic`: connects the current message to a recent message from the same speaker when both mention the same topic, capturing local topical continuity;

- `temporal_next`: connects each message to the immediately preceding message in the dialogue timeline and, separately, to the preceding message from the same speaker, providing lightweight temporal structure;
- `speaker_related`: connects adjacent messages from the same speaker to preserve speaker-specific continuity.

Persona snapshots are also indexed as fact or attribute nodes so that long-term user descriptors remain reachable from graph retrieval.

C.2 Graph Retrieval

Given a question, AdaMem first selects semantic seed nodes and then performs bounded multi-hop expansion over typed edges. Expansion can be filtered by the inferred target referent, so that user-focused questions remain centered on user-grounded evidence while ambiguous questions can access both bundles. In the current implementation, propagation uses lightweight type-specific edge weights rather than learned graph reasoning; therefore, the graph should be understood as a structured retrieval scaffold rather than a fully learned knowledge-graph reasoner.

D Case Studies

We analyze one representative success case in Figure 4 and one representative failure case in Figure 5 to better understand where AdaMem gains its advantage and where important challenges remain.

Question: What activity did Caroline used to do with her dad?

Conversation Excerpt:

```
Melanie: Oliver's hilarious! He hid his bone in
my slipper once! Cute, right? Almost as silly
as when I got to feed a horse a carrot.
Caroline: That's so funny! I used to go
horseback riding with my dad when I was a kid.
We'd go through the fields, feeling the wind.
It was so special. I've always had a love for
horses!
Melanie: Wow, that sounds great – I agree,
they're awesome. Here's a photo of my horse
painting I did recently.
```

Model Answers:

- **Mem0:** No information available ✗
- **AdaMem:** horseback riding ✓

Reference Answer: Horseback riding

Figure 4: **Case One (Success).**

Why AdaMem succeeds. This case is deceptively simple: although the answer span appears explicitly in the dialogue, the question is not a pure surface-form lookup. The system must first identify that the target speaker is *Caroline* rather than Melanie, then isolate the event embedded inside a broader horse-related exchange, and finally map the abstract query phrase "what activity" to the concrete experience "go horseback riding with my dad." Mem0 fails because flat semantic storage is easily distracted by nearby but less relevant horse-related details (e.g., Melanie feeding a horse or showing a horse painting), and it lacks an explicit mechanism to prioritize the participant-specific evidence most aligned with the question. By contrast, AdaMem benefits from its participant-aware memory organization and normalized message writing: the utterance can be stored as a structured event/fact associated with Caroline, her childhood, and the father-related activity. During inference, target resolution narrows retrieval to Caroline's memory bundle, while topic-to-message recovery and relation-aware evidence aggregation restore the exact supporting utterance instead of only a vague semantic neighbor. The case therefore illustrates the central advantage of AdaMem: it succeeds not merely by retrieving similar text, but by preserving who said what, under which personal context, and how the event should be reconstructed for answering.

Why AdaMem still fails. The failure case exposes a harder form of long-horizon reasoning that is only partially addressed by our current design. To answer correctly, the model must solve two implicit alignment problems: it must link the title mention "*nothing is impossible*" in the question to Melanie's deictic expression "this book" in the dialogue, and it must further normalize the relative temporal phrase "last year" into the absolute answer "2022." If either link is missing, retrieval returns incomplete evidence and the final answer becomes ungrounded. AdaMem improves temporal reasoning substantially in the aggregate, but its current pipeline still relies mainly on retrieval planning, graph expansion, and lightweight temporal salience signals rather than explicit canonicalization of relative time expressions at write time. As a result, the memory may preserve a coarse fact

Question: When did Melanie read the book "nothing is impossible"?

Conversation Excerpt:

```
Melanie: Wow, Caroline! You're so inspiring for
wanting to help others with their mental health.
What's pushing you to keep going forward with
it?
Caroline: I struggled with mental health, and
the support I got was really helpful. It made me
realize how important it is for others to have
a support system. So, I started looking into
counseling and mental health career options so
I could help other people on their own journeys
like I was helped.
Melanie: Caroline, so glad you got the support!
Your experience really brought you to where you
need to be. You're gonna make a huge difference!
This book I read last year reminds me to always
pursue my dreams, just like you are doing!
Caroline: Thanks so much, Mel! Seeing this pic
just made me appreciate my love of reading even
more. Books guide me, motivate me and help me
discover who I am. They're a huge part of my
journey, and this one's reminding me to keep
going and never give up!
```

Model Answers:

- **Mem0:** No evidence in memories ✗
- **AdaMem:** No evidence available ✗

Reference Answer: 2022

Figure 5: **Case Two (Failure).**

such as Melanie having read an inspiring book "last year," yet fail to store a stable symbolic binding between the book title and the absolute year required by the question. Once that canonical link is absent from memory, the Research Agent cannot recover it through additional search alone, and the Working Agent correctly defaults to abstention. This example suggests that the next bottleneck for AdaMem is not broad evidence coverage, but finer-grained temporal grounding and implicit entity linking, which is also consistent with the limitations discussed in the Limitations section.

E Prompt Templates

We provide the most representative prompt templates used in AdaMem.

E.1 Message Understanding Prompt

Role. This prompt is the entry point of the Memory Agent. It converts each incoming utterance into a normalized memory record with topic, attitude, reason, facts, attributes, and a short summary, which is then written into working memory and used by downstream memory update modules.

Content.

```
Perform topic tagging on this message from user.

Requirements:
1. Identify one primary topic/event in the
↪  message.
2. Infer the author's attitude toward the event.
3. Infer the reason behind the attitude.
4. Extract facts or events revealed by the
↪  message.
5. Extract user attributes revealed by the
↪  message.
6. Produce a one-sentence summary and a brief
↪  rationale.

Return JSON:
{
  "text": "original message",
  "tags": {
    "topic": ["..."],
    "attitude": ["Positive|Negative|Mixed"],
    "reason": ["..."],
    "facts": ["..."],
    "attributes": ["..."]
  },
  "summary": "...",
  "rationale": "..."
}

Message:
"{message}"
```

E.2 Episodic Memory Router Prompts

Role. After message understanding, AdaMem uses three router prompts to update long-term

episodic memory for *events/topics*, *facts*, and *attributes*. These prompts determine whether a new memory item should be added, used to update an existing record, or ignored as redundant.

Content.

```
You are a user profile updater that maintains a
↪  dictionary of existing
memory items. For each new message-derived item,
↪  analyze whether to update
the profile and respond in JSON format.

Input:
- Profile: existing topics / facts / attributes
- Message: a new structured item extracted from
↪  the latest utterance

Processing:
- Compare the new item with existing entries
↪  semantically.
- Choose one action:
  * UPDATE: a contradictory or revised entry
  ↪  already exists
  * ADD: the item is new
  * IGNORE: an equivalent entry already exists
- If there is no matched target, return "None".

Return JSON:
{
  "Action": "[UPDATE|ADD|IGNORE]",
  "Target": "matched entry or 'None'"
}
```

E.3 Topic Merge Prompt

Role. This prompt groups multiple fine-grained event topics into higher-level topic-centric memories. It is important for compressing long interaction histories into reusable themes without discarding event semantics.

Content.

```
Given a group of topics extracted from users'
↪  messages, analyze whether
these topics can be logically grouped under
↪  common themes.

Rules:
1. Only merge topics that talk about the same
↪  underlying event/theme.
2. Preserve the original meaning and context of
↪  each topic.
3. Extract as many common details as possible
↪  when naming the merged topic.
4. Do not reveal the user's attitude in the
↪  merged topic name.
5. Keep distinct concepts separate.

Input:
{
  "topic1": "summary sentence 1",
  "topic2": "summary sentence 2",
  ...
}

Return JSON:
{
  "Grouped Topics": {
    "NewTopicName1": ["original_topic1",
    ↪  "original_topic2"],
    "NewTopicName2": ["original_topic3"]
  },
  "Grouping Rationale": "Explanation of the
  ↪  grouping"
}
```

E.4 Route Refinement Prompt

Role. This prompt is used by the optional route refiner after the rule-based planner proposes an initial retrieval plan. Its purpose is not to freely redesign the retrieval pipeline, but to make a conservative adjustment to graph usage, hop depth, seed count, and fusion weights while avoiding unnecessary graph expansion for simple questions.

Content.

```
Refine a memory retrieval route plan for precise
↪  QA.
Prefer exact extraction for single-hop/temporal
↪  questions and avoid
unnecessary graph expansion.

Question:
{question}

Current plan JSON:
{current_plan}

Return only JSON with keys:
{
  "use_graph": true/false,
  "use_baseline": true/false,
  "graph_topn": integer,
  "hop_k": integer,
  "fusion_alpha": number,
  "fusion_beta": number,
  "fusion_gamma": number,
  "fusion_delta": number,
  "confidence": number
}
```

E.5 Research Integration and Reflection Prompts

Role. These prompts implement the core loop of the Research Agent. The integration prompt merges newly retrieved evidence into a consolidated working summary, while the reflection prompts judge whether the current summary is sufficient and, if not, generate follow-up retrieval requests.

Content.

```
[Integrate Prompt]
You are the IntegrateAgent. Merge newly retrieved
↪  evidence with the current
working notes to produce a consolidated factual
↪  summary relevant to the
QUESTION.
```

```
QUESTION:
{question}

EVIDENCE:
{evidence}

CURRENT RESULT:
{current_result}

Instructions:
1. Keep useful, on-topic information from CURRENT
↪  RESULT.
2. Add new, relevant, well-supported facts from
↪  EVIDENCE.
3. Remove off-topic content.
4. Prefer concrete details such as entities,
↪  dates, numbers, and events.
5. Resolve contradictions by preferring more
↪  specific or more recent evidence.
6. Use timestamps when answering temporal
↪  questions.

Return JSON:
{
  "content": "merged factual summary",
  "sources": ["source-1", "source-2", ...]
}
```

```
[Info Check Prompt]
You are the InfoCheckAgent. Judge whether the
↪  collected information is
sufficient to answer the QUESTION.

QUESTION:
{question}

RESULT:
{result}

Return JSON:
{
  "enough": true | false
}

[Follow-up Request Prompt]
You are the FollowUpRequestAgent. Generate
↪  focused follow-up retrieval
queries for the missing information.

QUESTION:
{question}

RESULT:
{result}

Instructions:
1. Identify what is still missing.
2. Generate 1-3 targeted retrieval queries.
3. Mention concrete entities or events whenever
↪  possible.

Return JSON:
{
  "new_requests": ["query-1", "query-2", ...]
}
```

E.6 Working Agent Answer Prompt

Role. This prompt is used by the Working Agent to convert the research summary into the final answer. It explicitly encourages concise, entity-centric responses and injects additional persona attributes or factual snippets when available.

Content.

```
[System Prompt]
You are role-playing as {speaker_b} in a
↪  conversation with {speaker_a}.
Your task is to answer questions about
↪  {speaker_a} or {speaker_b} in an
extremely concise manner based on the provided
↪  research summary.
Any content referring to 'User' refers to
↪  {speaker_a}.
Answer as concisely as possible and try to deduce
↪  clear answers rather than
return vague statements.

[User Prompt]
<RESEARCH SUMMARY>
{research_summary}

{extra_context}

The question is: {question}

Please only provide the content of the answer.
For date questions, use a specific format such as
↪  "15 July 2023" whenever
possible. For duration questions, answer in
↪  years, months, or days.
Generate answers primarily composed of concrete
↪  entities.
```