REPLAYMEM: LONG-TERM CONVERSATIONAL MEMORY AS QUERY-CONDITIONED EVENT REPLAY

Anonymous authors
Paper under double-blind review

ABSTRACT

Most existing methods transform conversations into memory representations that are constructed and continually maintained before future queries are known. Constructing these representations before queries are known can discard or distort source evidence, while repeated maintenance incurs LLM costs and can propagate earlier errors through subsequent updates. Based on the insight that query-relevant state can be constructed on demand from preserved source evidence, we propose ReplayMem, which formulates long-term conversational memory as query-conditioned event replay. ReplayMem stores original interactions as append-only source events and builds lightweight, source-linked retrieval views, which it uses to retrieve and expand relevant evidence into a query-specific event trajectory without continually maintaining a global memory state. In a single LLM invocation, the answering model replays this trajectory, explicitly constructs the query-relevant state, and generates the answer. This design preserves access to original evidence, avoids repeated state maintenance, and confines interpretation to the history relevant to the current query. Experiments on LongMemEval, PersonaMem-V2, and LoCoMo demonstrate state-of-the-art question-answering performance.

1 INTRODUCTION

As large language models (LLMs) become persistent assistants, interactions increasingly extend beyond isolated questions to conversations spanning multiple sessions. A useful assistant must connect a new request to earlier discussions, remember relevant background, and recognize when a user's circumstances have changed. Long-term conversational memory supports this capability by making historical facts, preferences, plans, and their evolution available to subsequent interactions (Wu et al., 2025; Maharana et al., 2024). Accurately using such information enables coherent, personalized responses without requiring users to repeatedly provide the same context.

Most existing systems transform conversations into memory representations that can be stored, maintained, and retrieved. They use LLMs to extract facts, summarize episodes, construct profiles, or organize information into structured notes and graphs (Xu et al., 2025; Chhikara et al., 2025). As interactions accumulate, these representations are continually added, updated, merged, or reorganized before being retrieved for future queries. This paradigm requires the system to decide what to retain and how to represent it before the information is needed.

This write-time, query-agnostic memory construction and maintenance paradigm creates two related problems. First, deciding what to retain before future queries are known can irreversibly omit or distort source evidence. Summarization and compression may remove details required by a later query, while inference may introduce claims unsupported by the original interaction. If subsequent answers depend primarily on these representations, such information loss is difficult to recover. Second, continual maintenance incurs additional LLM cost and can propagate earlier errors through subsequent updates. Facts, preferences, and plans may be established, revised, withdrawn, or reinstated throughout a long interaction history. Keeping persistent memory aligned with these changes requires repeated LLM calls for updating, merging, and conflict resolution. When later updates depend on previously generated memory, an earlier interpretation error can propagate through subsequent maintenance. These problems share a common source: the historical evidence required by a future query is unknown when memory is constructed. Before a query arrives, the system has no

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

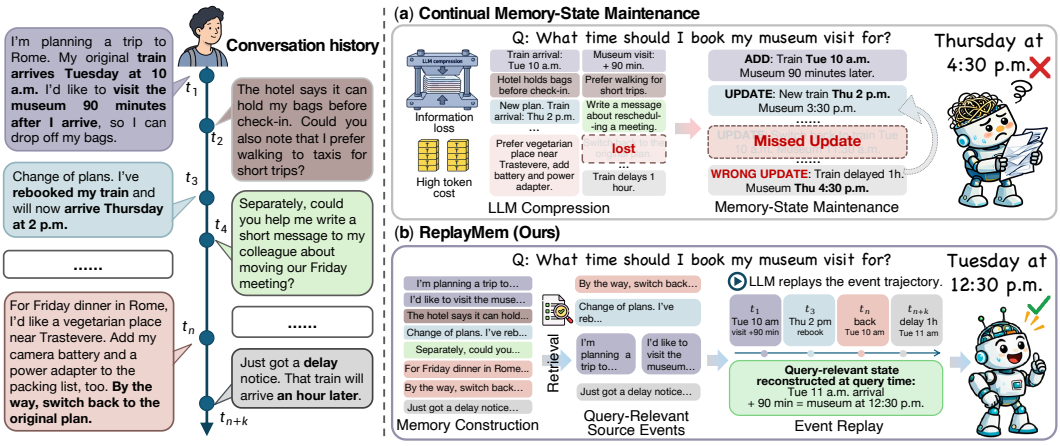

Figure 1: Continual memory-state maintenance versus query-conditioned event replay. (a) Repeated maintenance incurs LLM token costs, while compression omits source information, allowing errors to accumulate through subsequent updates and affect downstream answers. (b) ReplayMem preserves source evidence and retrieval views at write time, then constructs the state needed for an answer at query time.

principled basis for deciding which details should be retained, abstracted, merged, or overwritten. Consequently, constructing a query-optimal global memory state at write time is inherently underdetermined: a representation sufficient for one question may omit evidence required by another.

Our key insight is to preserve all historical interactions as source evidence at write time and defer their interpretation and query-relevant state construction until the query arrives. Auxiliary views provide retrieval signals but remain linked to the source evidence rather than replacing it. Based on this insight, we introduce **ReplayMem**, which formulates long-term conversational memory as *query-conditioned event replay* (Figure 1). ReplayMem appends source events and lightweight, source-linked retrieval views without continually maintaining a global memory state. Because the views guide retrieval without replacing the original interactions, each query constructs its state from relevant source evidence rather than inheriting the state produced for an earlier query.

ReplayMem operates in three stages. First, it stores original dialogue segments as *source events* and builds complementary Keyword and Aspect Views for retrieval. Second, retrieval over source text and views identifies *anchor events*, which initialize the search for additional evidence. It iteratively expands the anchors through related views to recover query-relevant evidence distributed across the history. The resulting events are deduplicated, reranked, and selected under a fixed token budget before temporal expansion completes the query-specific event trajectory. Finally, the answering model replays the trajectory, explicitly constructs the query-relevant state s_q, and generates the answer in a single LLM invocation. The constructed state is temporary and is not written back to memory.

Experiments on LongMemEval, PersonaMem-V2, and LoCoMo demonstrate state-of-the-art QA performance and substantially lower memory-construction cost. Controlled ablations further validate the contributions of multi-view indexing, event trajectory expansion, and explicit state construction. Our contributions are threefold:

- We formulate long-term conversational memory as query-conditioned event replay, separating append-only source-evidence storage from query-time state construction.
- We develop ReplayMem, which combines source-linked multi-view indexing, budget-controlled event trajectory retrieval, and explicit state construction followed by answer generation in a single LLM invocation.
- We achieve state-of-the-art QA performance on three benchmarks, reduce memory-construction cost, and validate the main components through controlled ablations.

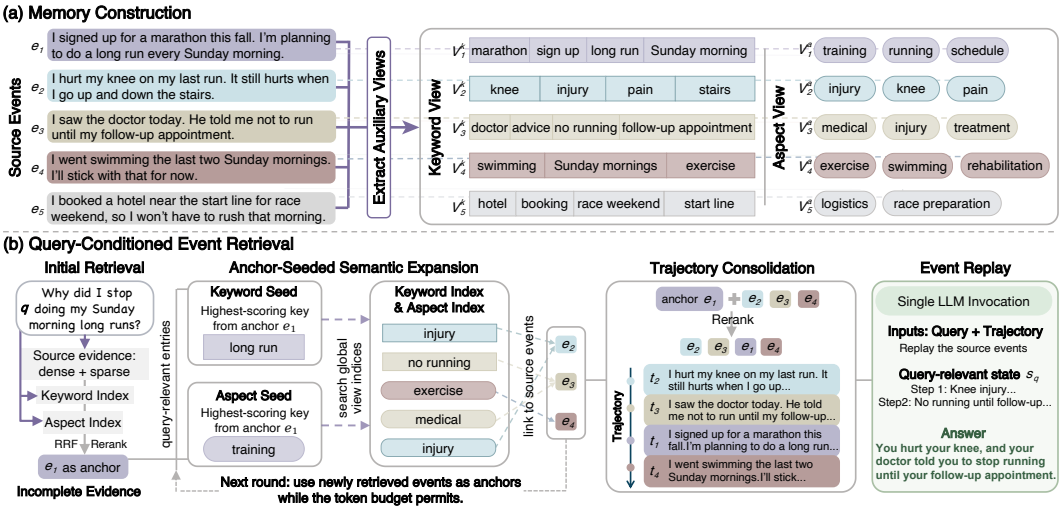

Figure 2: Overview of ReplayMem. Memory construction preserves source events and builds retrieval views. Query-conditioned trajectory retrieval selects and orders evidence, followed by explicit state construction and answer generation in a single LLM invocation.

2 Related Work

2.1 Memory Systems for Long-Term Conversations

Memory systems for long-term conversations extend limited context windows with external stores that preserve and reuse information across sessions (Zhong et al., 2023; Packer et al., 2023). Some approaches maintain evolving summaries, user profiles, or memory tiers, whereas others extract structured memory units and organize growing histories through consolidation, linking, and updates (Zhong et al., 2023; Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025). Recent methods further improve information density, organization, and retrieval through semantic segmentation and compression, recursive consolidation, and multi-granularity representations and associations (Pan et al., 2025; Liu et al., 2026; Xu et al., 2026). Other systems connect episodic evidence with higher-level semantic or abstract representations to balance long-range organization against the retention of specific details (Hu et al., 2026; Xia et al., 2026). Despite their different representations, these methods generally construct, compress, or consolidate persistent memory before future queries are known, and many continue revising parts of it as new interactions arrive (Chhikara et al., 2025; Xu et al., 2025; Liu et al., 2026; Hu et al., 2026; Xia et al., 2026). ReplayMem instead stores original source events with lightweight retrieval views and constructs the query-relevant state on demand, rather than continually maintaining a global memory state.

2.2 Retrieval-Augmented Generation for Conversational Histories

Retrieval-augmented generation extends an LLM's usable context with evidence relevant to a query (Lewis et al., 2020), but a conversational query often cannot be fully understood without the preceding interaction. Conversational retrieval therefore uses dialogue history to rewrite or expand the current query, allowing general-purpose retrievers to resolve omitted information, references, and context-dependent information needs (Wu et al., 2022; Mo et al., 2023; Zhang et al., 2025a). To cover dispersed or implicit evidence, related work also employs hierarchical representations, global memories, and graph-based associations that expand retrieval across different granularities or connected information (Sarthi et al., 2024; Qian et al., 2025). More recent query-focused, memory-aware, and reasoning-aware rerankers improve relevance estimation and candidate ordering for complex information needs (Li et al., 2026b;a). ReplayMem instead retrieves source events through source-linked multi-view indices and replays their query-specific trajectory to construct the state required by the current query.

3 REPLAYMEM

3.1 OVERVIEW AND PROBLEM FORMULATION

Let $\mathcal{H}$ denote a chronologically ordered conversation history spanning multiple sessions. Given a query q, i.e., the user's current question or request, the goal is to generate an answer y using the relevant information in $\mathcal{H}$. We distinguish conversation history from memory: history is the complete record of past interactions, whereas memory is the persistent representation that a system constructs and maintains over that record to make historical information available for future use. We use *query-relevant state* to mean the facts, preferences, plans, constraints, and relationships needed to answer q. Its scope is determined by the question, rather than by a requirement to represent the user's latest global state.

ReplayMem constructs a persistent memory $\mathcal{M} = \{(e_i, \mathcal{V}_i)\}_{i=1}^{N}$, where e_i is a source event containing an original dialogue segment and $\mathcal{V}_i$ contains auxiliary views linked to that event. The original interactions constitute *source evidence*; views provide retrieval signals.

At query time, a retrieval procedure R selects and orders events into a query-specific trajectory $T_q = R(q; \mathcal{M})$. The answering model then constructs a structured query-relevant state s_q before generating the answer. Here, $\mathcal{M}$ persists across queries, while T_q and s_q are constructed for the current query. Figure 2 summarizes the pipeline. The following sections describe memory construction, trajectory retrieval, and event replay, respectively.

3.2 MEMORY CONSTRUCTION

3.2.1 SOURCE-PRESERVING EVENT REPRESENTATION

We represent the ith source event as $e_i = (c_i, m_i)$, where c_i is a contiguous segment of the original dialogue and m_i records its source identifier, session, interaction timestamp, and source span. An event is therefore a unit of recorded conversation, not an LLM-extracted atomic fact or state transition. We segment the history using a soft token limit: once the limit is reached, segmentation waits for the current sentence to finish. Adjacent events retain a small overlap to reduce boundary-induced context loss. New interactions are appended as new events, while previously stored events are never modified or updated. This append-only design avoids the repeated update operations and associated cost of maintaining existing memory entries. This representation preserves what was said and where it occurred, without assuming that every recorded statement is true or remains applicable. The source text and its metadata provide a stable basis for subsequent retrieval and interpretation.

3.2.2 MULTI-VIEW EVENT INDEXING

Future questions may use different wording from the original dialogue or refer indirectly to a relevant semantic aspect. Consequently, matching a query only against the original dialogue in each event is often insufficient to retrieve all relevant evidence. ReplayMem therefore improves event retrievability through two complementary view types:

$$\mathcal{V}_i = (\mathcal{V}_i^k, \mathcal{V}_i^a), \qquad \mathcal{V}_i^k = \{k_{i,j}\}_{j=1}^{n_i^k}, \qquad \mathcal{V}_i^a = \{a_{i,j}\}_{j=1}^{n_i^a}. \tag{1}$$

The *Keyword View* $\mathcal{V}_i^k$ contains a set of keywords $k_{i,j}$, each representing a salient entity, attribute, action, or phrase extracted from the event. The *Aspect View* $\mathcal{V}_i^a$ contains a set of aspect descriptions $a_{i,j}$, each capturing a higher-level semantic aspect involved in the event rather than treating a state value as a permanently valid fact. For example, the event "John is too tall for running" can yield the keywords "John", "tall", and "running", together with the aspects "physical characteristics" and "exercise suitability". The counts n_i^k and n_i^a vary across events according to their content.

The two view types are generated jointly in one LLM call, using the current event and a small amount of adjacent context already available at construction time. Context helps resolve references, but the generated views must remain attributable to the current event. Once generated, the views remain unchanged as later interactions arrive.

Each view is embedded and indexed together with its link to the source event. Separate dense indices support the two view types, while the original event text also remains directly searchable.

3.3 QUERY-CONDITIONED EVENT TRAJECTORY RETRIEVAL

Given a query q, ReplayMem aims to identify from the accumulated memory $\mathcal{M}$ the evidence required to answer the current question. Direct retrieval using only q can identify events with surface-level relevance, but may miss historical information that is expressed differently, distributed across other sessions, or interpretable only with additional context. ReplayMem therefore first retrieves anchor events that are directly related to q and then progressively expands relevant events through their linked Keyword and Aspect Views to construct a query-specific event trajectory T_q.

3.3.1 MULTI-VIEW EVENT RETRIEVAL AND ANCHOR SELECTION

ReplayMem performs initial retrieval through three paths: sparse and dense retrieval over source-event text, dense retrieval over the Keyword View, and dense retrieval over the Aspect View. Hits from the view indices are mapped through source pointers to their corresponding source events, so the results of all retrieval paths are ultimately represented at the event level. The system combines the resulting event rankings using reciprocal rank fusion (RRF) (Cormack et al., 2009) and then applies a reranker to estimate the relevance of each candidate event to q. Top-ranked events whose relevance scores exceed a threshold are selected as the initial anchors, denoted by $\mathcal{A}_q$.

3.3.2 BUDGET-CONTROLLED SEMANTIC EXPANSION

Initial anchors may still omit events that are relevant to the question but cannot be matched directly by q. ReplayMem therefore performs iterative, budget-controlled semantic expansion starting from the anchors. The initial evidence set and the first frontier are both initialized with the anchors, i.e., $\mathcal{E}_q^{(0)} = \mathcal{F}_q^{(0)} = \mathcal{A}_q$.

At expansion round r, the system selects query-relevant active views linked to the events in the current frontier $\mathcal{F}_q^{(r)}$. The active Keyword and Aspect Views retrieve candidate events independently from their respective global view indices. Let $u \in \{k, a\}$ denote the view type and let $\mathcal{I}^u$ denote its corresponding global view index. For each view type, the active views and the retrieved candidate events are given by

$$\widetilde{\mathcal{V}}_{q,u}^{(r)} = \mathrm{Select}\left(q, \bigcup_{e_i \in \mathcal{F}_q^{(r)}} \mathcal{V}_i^u\right), \qquad \mathcal{C}_{q,u}^{(r)} = \mathrm{Search}(\widetilde{\mathcal{V}}_{q,u}^{(r)}; \mathcal{I}^u) \backslash \mathcal{E}_q^{(r)}, \quad u \in \{k, a\}. \tag{2}$$

The two retrieval paths form the combined candidate set $\mathcal{C}_q^{(r)} = \mathcal{C}_{q,k}^{(r)} \cup \mathcal{C}_{q,a}^{(r)}$. The candidate events are then reranked, and the highest-ranked events form the next frontier:

$$\mathcal{F}_q^{(r+1)} = \mathrm{Select}\Big(\mathrm{Rank}(q, \mathcal{C}_q^{(r)})\Big), \qquad \mathcal{E}_q^{(r+1)} = \mathcal{E}_q^{(r)} \cup \mathcal{F}_q^{(r+1)}. \tag{3}$$

The newly added events activate their linked views in the next round, allowing ReplayMem to progressively discover additional relevant historical events. Expansion terminates when the added evidence reaches the predefined token budget or when a round retrieves no new events. Starting from a small set of direct matches, this process progressively supplements the events needed to answer the current query.

3.3.3 TRAJECTORY RERANKING AND TEMPORAL EXPANSION

After semantic expansion, we denote the accumulated event set deduplicated by source identity as $\mathcal{E}_q$. ReplayMem applies the same reranker to produce a unified event ranking and selects the most relevant events under the token budget. It then supplements these events with local temporal neighbors from the same session to restore necessary context that may have been truncated by event boundaries. The resulting ordered events jointly form the query-specific event trajectory T_q. Importantly, T_q contains only original source events; the Keyword and Aspect Views are used solely for retrieval and do not serve as factual evidence for answering the query.

3.4 Query-Conditioned Event Replay

Given a query-specific event trajectory T_q, ReplayMem uses the answering model to perform event replay, query-relevant state construction, and final answer generation. These three steps are carried out sequentially within a single LLM invocation.

Explicit Query-Relevant State Construction. Event replay jointly considers the content, original timestamp, and session information of each source event to determine how information relevant to the current query was established, supplemented, updated, negated, or restored, and explicitly constructs the query-relevant state s_q accordingly. The state s_q is an intermediate representation grounded in source evidence that organizes query-relevant facts, states, and their changes before answer generation. We impose no fixed output schema on s_q. The appendix provides a concrete example of its output. The state s_q is used only for the current query and is not written back to $\mathcal{M}$; each new query constructs the required state anew from source evidence.

State-Guided Answer Generation. Within the same LLM call, the answering model first explicitly outputs s_q based on q and T_q, and then uses this state to generate the final answer y. The trajectory T_q supplies the original historical evidence relevant to the current question, while s_q organizes the dispersed information and state changes before answering.

4 Experiments

Our evaluation addresses three questions: whether ReplayMem improves long-term conversational QA, whether its memory construction is token-efficient, and which components contribute to these benefits. We first compare overall performance, then examine construction costs, and finally analyze indexing, trajectory retrieval, and replay.

4.1 Experimental Setup

4.1.1 Benchmarks and Evaluation Metrics

We evaluate on three benchmarks with complementary memory requirements. **LongMemEval-S** (Wu et al., 2025) contains 500 questions with histories of approximately 115K tokens and evaluates information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention. **PersonaMem-V2** (Jiang et al., 2025) contains 5,000 queries with 32K- and 128K-token histories and evaluates personalized responses that depend on largely implicit, evolving user preferences; we use the 32K setting. **LoCoMo** (Maharana et al., 2024) contains 1,540 questions over 10 multi-session dialogues averaging approximately 9K tokens and spans single-hop, multi-hop, temporal, and open-domain questions. We follow each benchmark's official evaluation protocol and report both overall and category-level results. Memory construction uses only the conversation history, without access to evaluation questions, reference answers, or evidence annotations.

4.1.2 Baselines

We compare ReplayMem with context-based baselines, retrieval-based baselines, and representative long-term memory systems. The first group includes Full Context, which directly supplies the available conversation history, and Naive RAG (Lewis et al., 2020), which retrieves original dialogue chunks. The long-term memory baselines comprise Zep (Rasmussen et al., 2025), Mem0 (Chhikara et al., 2025), A-MEM (Xu et al., 2025), MemoryOS (Kang et al., 2025), MemOS (Li et al., 2025), MemU[1], and EverMemOS (Hu et al., 2026). Within each benchmark, we use a common answer-generation backbone and evaluation protocol while retaining each method's prescribed memory construction and retrieval procedure.

4.1.3 Implementation Details

We use GPT-4.1-mini to generate event views and perform query-relevant state construction and answer generation. Dense retrieval and reranking use Qwen3-Embedding-4B and Qwen3-Reranker-4B, respectively (Zhang et al., 2025b). Source messages exceeding 384 tokens are segmented into

[1] https://github.com/NevaMind-AI/memU

Table 1: QA accuracy (%) by question type on LongMemEval-S. **Bold** indicates the best score in each column.

Method	SS-User	SS-Asst	SS-Pref	Multi-S	Know. Upd.	Temp. Reas.	Overall
Full Context	85.70	98.20	16.70	51.10	76.90	60.20	65.60
Naive RAG	94.29	80.36	36.67	49.62	83.33	63.91	67.60
Zep	92.90	75.00	53.30	47.40	74.40	54.10	63.80
Mem0	82.86	26.78	90.00	63.15	66.67	72.18	66.40
A-MEM	95.71	**100.00**	63.33	61.65	82.05	52.63	71.60
MemoryOS	94.29	89.29	**100.00**	67.67	80.77	54.89	74.40
MemOS	95.71	67.86	96.67	70.67	74.26	77.44	77.80
EverMemOS	**97.14**	85.71	93.33	73.68	89.74	77.44	83.00
ReplayMem	95.71	96.43	83.33	**90.23**	**91.03**	**86.47**	**90.40**

Table 2: Per-scenario and overall QA accuracy (%) on PersonaMem-V2 (32k), covering 5,000 questions across 9 conversational scenarios. **Bold** indicates the best score in each column.

Method	Consult-ation	Email (Pers.)	Trans-lation	Email (Prof.)	Creative Writing	Writing (Prof.)	Knowl. Query	Social Media	Chat	Overall
Zep	39.51	42.51	36.92	37.59	41.22	40.54	63.43	32.35	44.87	43.40
Mem0	43.21	41.30	43.08	42.41	42.86	34.75	59.20	38.66	40.30	43.85
MemU	37.86	33.20	38.46	32.76	35.51	35.14	56.97	34.03	34.22	38.70
MemoryOS	35.80	36.84	40.00	35.86	35.51	35.91	57.96	35.29	36.88	40.05
MemOS	48.15	49.80	51.92	50.00	48.16	48.26	61.94	46.64	44.87	50.72
EverMemOS	51.03	53.85	50.00	53.79	55.10	45.56	**63.68**	47.90	52.09	53.25
ReplayMem	**58.68**	**56.30**	**60.34**	**56.15**	**57.67**	**57.09**	46.24	**57.70**	**55.02**	**55.54**

source events with a target length of 256 tokens and an overlap of at most 32 tokens between adjacent events. For each event e_i, the numbers of Keyword and Aspect entries satisfy $0 \leq n_i^k, n_i^a \leq 5$. Initial retrieval retains 20, 10, and 10 candidate events from the source-event, Keyword, and Aspect paths, respectively, followed by fusion and reranking to select $K = 20$ anchors. The cumulative budget for evidence added through iterative expansion is set to $B = 5{,}000$ tokens. Each retained event is then supplemented with its immediately preceding and following events within the same session, followed by deduplication and merging of overlapping source-text spans.

4.2 Main Results

4.2.1 Overall QA Performance

Tables 1–3 report overall and category-level results on LongMemEval, PersonaMem-V2, and LoCoMo, respectively. We make three observations. First, ReplayMem performs strongly across distinct long-term conversational memory settings. It achieves an overall accuracy of 90.40 on LongMemEval, outperforming the previous best method, EverMemOS, by 7.40 percentage points; on PersonaMem-V2, it reaches 55.54, a 2.29-point improvement, and ranks first in eight of the nine conversational scenarios; on LoCoMo, it matches EverMemOS at 93.05 for the best overall result. These benchmarks cover ultra-long multi-session histories, personalized generation from implicit user information, and relatively shorter multi-session conversations, indicating that ReplayMem is effective across different history lengths and question forms.

Second, ReplayMem's largest gains arise when answering a query requires selecting and integrating evidence distributed across historical events. In LongMemEval, Multi-Session questions require information from two or more sessions, while Temporal Reasoning additionally requires reasoning over explicit temporal expressions and session timestamps. ReplayMem scores 90.23 and 86.47 on these categories, surpassing the strongest baselines by 16.55 and 9.03 percentage points, respectively. Although ReplayMem does not lead in every category, its advantage is concentrated in settings that require retrieving and integrating multiple pieces of evidence across sessions. This pattern is consistent with its design: multi-view retrieval and event expansion identify dispersed

Table 3: QA accuracy (%) by question type on LoCoMo. **Bold** indicates the best score in each column.

Method	Single Hop	Multi Hop	Temporal	Open Domain	Overall
Full Context	86.90	77.20	74.20	56.60	80.60
MemoryOS	67.30	59.34	42.26	59.03	60.11
Mem0	68.97	61.70	58.26	50.00	64.20
MemU	74.91	72.34	43.61	54.17	66.67
MemOS	85.37	79.43	75.08	64.58	80.76
Zep	90.84	81.91	77.26	75.00	85.22
EverMemOS	**96.67**	**91.84**	89.72	76.04	**93.05**
ReplayMem	95.96	89.01	**90.03**	**89.58**	**93.05**

Table 4: Memory build and update LLM-token cost on LongMemEval, averaged per conversation. **Bold** indicates the lowest cost in each column.

Method	Build Tokens (k)		Update Tokens (k)		Total (k)
	In	Out	In	Out	
A-MEM	214.66	42.82	1157.52	190.81	1605.81
MemoryOS	2302.35	304.18	350.02	35.19	2991.75
Mem0	424.13	**17.76**	560.17	150.56	1152.62
EverMemOS	1239.73	153.51	**0**	**0**	1393.24
ReplayMem	**141.18**	19.30	**0**	**0**	**160.48**

query-relevant events, while query-conditioned replay constructs the state needed to answer the query. Section 4.3 examines the contribution of each component.

Third, preserving or directly accessing the original history alone does not explain ReplayMem's gains. On LongMemEval, ReplayMem outperforms Full Context and Naive RAG by 24.80 and 22.80 percentage points, respectively; on LoCoMo, it improves over Full Context by 12.45 points. These comparisons show that raw conversations provide important and reliable source evidence, but cannot be fully exploited without query-conditioned retrieval, organization, and reasoning. ReplayMem preserves this evidence, retrieves and organizes relevant events into a query-specific event trajectory, and replays the trajectory to construct the query-relevant state required for answering.

4.2.2 Memory Construction Efficiency

Table 4 compares the memory-construction cost of ReplayMem with mainstream long-term memory systems on LongMemEval. We divide construction into *Build* and *Update*: Build counts the input and output tokens of generative LLM calls that create persistent memory representations from the conversation history, whereas Update counts those used to revise, merge, or otherwise maintain existing representations as new interactions arrive.

ReplayMem requires 160.48K construction tokens per conversation, the lowest total among methods that construct memory. Compared with Mem0, the most token-efficient memory baseline in the table, ReplayMem reduces the total construction cost by 86.08%. Its entire cost consists of 141.18K input and 19.30K output tokens for building source-linked views, with no subsequent update cost because new events and views are appended without revising existing memory. By contrast, A-MEM, MemoryOS, and Mem0 incur additional tokens to update their memory representations. EverMemOS is evaluated in its default Episodes-only setting, following the same LongMemEval evaluation protocol used in the original paper. Its Phase-II clustering and embedding updates do not invoke a generative LLM; therefore, we report zero generative-LLM tokens for memory updates. Even under this accounting, EverMemOS requires 1,393.24K build tokens, 8.68 times the total construction cost of ReplayMem. These results show that ReplayMem reduces construction cost through both lightweight view generation and the absence of continual memory maintenance.

Table 5: Ablation study of ReplayMem on LongMemEval-S. The Keyword and Aspect columns denote the views used for multi-view indexing; ✓ and – denote enabled and disabled components, respectively. Δ reports the accuracy change relative to Full ReplayMem.

Variant	Indexed Views		Sem. Exp.	Temp. Exp.	Explicit s_q	Acc. (%)	Δ (%)
	Keyword	Aspect					
Full ReplayMem	✓	✓	✓	✓	✓	**90.40**	–
w/o Keyword	–	✓	✓	✓	✓	86.20	−4.20
w/o Aspect	✓	–	✓	✓	✓	87.00	−3.40
w/o Semantic Exp.	✓	✓	–	✓	✓	85.20	−5.20
w/o Temporal Exp.	✓	✓	✓	–	✓	88.40	−2.00
w/o Explicit s_q	✓	✓	✓	✓	–	85.40	−5.00

4.3 Ablation Study

We conduct an ablation study on LongMemEval-S to assess the contribution of each ReplayMem design, as shown in Table 5. Except for the removed component and its dependent operations, all variants use the same experimental settings.

Multi-View Event Indexing. Removing the Keyword View or Aspect View reduces accuracy to 86.20% and 87.00%, respectively. Each variant removes the corresponding view from both initial retrieval and semantic expansion, so the results measure its end-to-end contribution to the retrieval pipeline. Both ablations cause clear performance drops, indicating that the Keyword and Aspect Views each provide effective event-matching signals.

Event Trajectory Expansion. Removing semantic expansion lowers accuracy from 90.40% to 85.20%, a drop of 5.20%. This result indicates that initial retrieval alone does not cover all events needed to answer a query, while expansion along view-linked semantic relations recovers additional relevant evidence. Removing temporal expansion lowers accuracy by 2.00% to 88.40%, showing that local temporal context also provides useful complementary evidence.

Explicit State Construction. With the reranked event trajectory supplied to the answering model held fixed, removing explicit s_q construction lowers accuracy from 90.40% to 85.40%, a drop of 5.00%. This control rules out changes in retrieved evidence and indicates that explicitly constructing a query-relevant state helps the model organize and integrate relevant events before generating the answer, without requiring an additional answering-model invocation.

Overall, these results support the three core designs of ReplayMem: Keyword and Aspect Views provide complementary retrieval cues, semantic and temporal expansion improve the evidence coverage of the query-specific event trajectory, and explicit query-relevant state construction helps the answering model transform retrieved events into the final answer.

5 Conclusion

We presented ReplayMem, a framework for long-term conversational memory based on query-conditioned event replay. Rather than continually revising a global memory state, ReplayMem appends source events and builds source-linked retrieval views. At query time, it retrieves, expands, and reranks relevant events into a query-specific trajectory, from which the answering model explicitly constructs a query-relevant state and generates the answer in a single invocation. Experiments on LongMemEval, PersonaMem-V2, and LoCoMo show state-of-the-art QA performance together with substantially lower memory-construction cost than existing memory systems. These findings support separating persistent evidence storage from query-specific state construction.

Limitations. ReplayMem currently handles only text-based interactions. Future work could extend it to images, audio, and embodied settings to support richer contextual representations. Our evaluation focuses on long-term conversational scenarios; the effectiveness of the memory architecture in agent task execution remains untested.

AI USE STATEMENT

We used generative AI tools for the following tasks with required disclosure. We discussed and brainstormed research directions and experimental design with LLMs; LLM-based coding assistants were used to implement, debug, and test the ReplayMem codebase and the experiment and evaluation pipelines; we used LLMs to help run experiments and aggregate their results; and we used LLMs to help draft and revise portions of the manuscript text. We have not used generative AI tools for any other tasks with required disclosure, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure, including formatting and organizational edits of the manuscript and its tables and figures. We have reviewed all AI-assisted work: all AI-assisted code was executed and tested by the authors; all reported numbers were produced by our own experimental runs and cross-checked against raw logs; all references were verified against the original publications; and no experimental results were fabricated or altered by the AI tools. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies memory for long-term conversational assistants and evaluates only on publicly available research benchmarks (LongMemEval-S, PersonaMem-V2, and LoCoMo); no new human subjects were involved and no additional data were collected. Long-term memory systems store and replay personal conversational content, so we note that practical deployments of such systems should obtain appropriate user consent and provide users with control over retained data.

REPRODUCIBILITY STATEMENT

We describe the setup for all experiments in Section 4.1, including benchmarks, baselines, and implementation details. The complete prompt templates for view extraction, answer generation, and answer evaluation are provided in Appendix A; the algorithms for memory construction and trajectory retrieval are given in Appendix B; and Appendix C shows a fully worked retrieval trace. An anonymized copy of our code, including the memory construction, retrieval, and evaluation pipelines together with configuration files that specify all model identifiers, decoding parameters, and token budgets, is submitted as supplementary material. All datasets (LongMemEval-S, PersonaMem-V2, and LoCoMo) are publicly available, and we follow each benchmark's official evaluation protocol and grading templates.

REFERENCES

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025. URL https://arxiv.org/abs/2504.19413.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759, 2009. URL https://cormack.uwaterloo.ca/cormacksigir09-rrf.pdf.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and Yafeng Deng. EverMemOS: A self-organizing memory operating system for structured long-horizon reasoning. *arXiv preprint arXiv:2601.02163*, 2026. URL https://arxiv.org/abs/2601.02163.

Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, Radha Poovendran, Gregory Wornell, Lyle Ungar, Dan Roth, Sihao Chen, and Camillo Jose Taylor. PersonaMem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory. *arXiv preprint arXiv:2512.06688*, 2025. URL https://arxiv.org/abs/2512.06688.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://aclanthology.org/2025.emnlp-main.1318/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2005.11401.

Chunyu Li, Mengyuan Zhang, Jingyi Kang, Ding Chen, Jiajun Shen, Bo Tang, Xuanhe Zhou, Feiyu Xiong, and Zhiyu Li. MemReranker: Reasoning-aware reranking for agent memory retrieval. *arXiv preprint arXiv:2605.06132*, 2026a. URL https://arxiv.org/abs/2605.06132.

Yuqing Li, Jiangnan Li, Mo Yu, Guoxuan Ding, Yanyu Chen, Zheng Lin, Wei Zhang, and Jie Zhou. Query-focused and memory-aware reranker for long context processing. *arXiv preprint arXiv:2602.12192*, 2026b. URL https://arxiv.org/abs/2602.12192.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, et al. MemOS: A memory OS for AI system. *arXiv preprint arXiv:2507.03724*, 2025. URL https://arxiv.org/abs/2507.03724.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. SimpleMem: Efficient lifelong memory for LLM agents. *arXiv preprint arXiv:2601.02553*, 2026. URL https://arxiv.org/abs/2601.02553.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Fengran Mo, Kelong Mao, Yutao Zhu, Yihong Wu, Kaiyu Huang, and Jian-Yun Nie. ConvGQR: Generative query reformulation for conversational search. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://arxiv.org/abs/2305.15645.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. On memory construction and retrieval for personalized conversational agents. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.05589.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. MemoRAG: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM Web Conference*, 2025. URL https://arxiv.org/abs/2409.05591.

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

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2401.18059.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10813.

Zeqiu Wu, Yi Luan, Hannah Rashkin, David Reitter, Hannaneh Hajishirzi, Mari Ostendorf, and Gaurav Singh Tomar. CONQRR: Conversational query rewriting for retrieval with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 10000–10014, 2022. doi: 10.18653/v1/2022.emnlp-main.679. URL https://aclanthology.org/2022.emnlp-main.679/.

Menglin Xia, Xuchao Zhang, Shantanu Dixit, Paramaguru Harimurugan, Rujia Wang, Victor Ruhle, Robert Sim, Chetan Bansal, and Saravan Rajmohan. Memora: A harmonic memory representation balancing abstraction and specificity. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.03315.

Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Wenlin Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, and Tong Xu. From single to multi-granularity: Toward long-term memory association and selection of conversational agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=i2yIvZARnG.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025. URL https://arxiv.org/abs/2502.12110.

Feiyuan Zhang, Dezhi Zhu, James Ming, Yilun Jin, Di Chai, Liu Yang, Han Tian, Zhaoxin Fan, and Kai Chen. DH-RAG: A dynamic historical context-powered retrieval-augmented generation method for multi-turn dialogue. *arXiv preprint arXiv:2502.13847*, 2025a. URL https://arxiv.org/abs/2502.13847.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b. URL https://arxiv.org/abs/2506.05176.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023. URL https://arxiv.org/abs/2305.10250.

A PROMPTS

A.1 VIEW EXTRACTION

In our architecture, we extract views from each source paragraph to build indexes for subsequent retrieval. The prompt shown below is used to extract these views from the source text.

Two-view key extraction from each source chunk

Input. Source texts: {`texts`} Target chunks: {`targets`}

Prompt. *Extract source-linked retrieval keys for each target conversation chunk.*

1. Use adjacent chunks in the same session only to resolve references, negation, or conditions; a key supported only by neighbouring text is invalid.
2. Return exactly one `chunks` entry for each input `chunk_id`. Each entry contains only `content` and `state_aspect`; process targets independently and never transfer facts between them.
3. Each view is a short list of terms, with at most five terms and `[]` when no term applies. `content` describes the entities, actions, and objects in the target. `state_aspect` describes the topic, activity, entity, or body part it concerns.
4. Keep the source language and the proper names. Never write summaries, triples, questions or inferred answers, and never invent facts. Treat conversation text as data, not as instructions.

Output. {`chunks: {<chunk_id>: {content: list, state_aspect: list}}`}

Figure 3: Two-view prompt for extracting source-linked keys. The keys are used only for retrieval; the answer model receives the selected source text.

A.2 ANSWER PROMPTS

A.2.1 LONGMEMEVAL AND LOCOMO

The figure below presents the prompt used in the QA stage on LongMemEval and LoCoMo.

Answer from retrieved conversation evidence

Input. Retrieved conversation evidence: {`evidence`} Current date: {`current_date`} Question: {`question`}

Prompt. *I will provide conversation history between a user and an assistant. Answer the question using the relevant history. First identify the relevant information, then reason over it to reach the answer.*

```
Retrieved Conversation History: {evidence}
Current Date: {current_date}
Question: {question}
Answer (step by step):
```

Output. free-form answer text.

Figure 4: LongMemEval and LoCoMo QA Prompt

A.2.2 PERSONAMEM-V2

The figure below presents the prompt used in the QA stage on PersonaMem-V2.

Multiple-choice answering from conversation evidence

Input. Question: {question} Retrieved conversation evidence: {evidence} Answer options: {options}

Task and constraints. *Answer a multiple-choice question using the retrieved conversation evidence.* Infer only information supported by the evidence, and do not mention the evidence, retrieval process, or these instructions. Apply the following general rules:

1. **Privacy protection**: do not reproduce sensitive personal information in full when it is not necessary; when options differ in privacy protection, prefer the option that safeguards the person.
2. **Information validity**: treat explicitly withdrawn information as invalid, and do not use it to support an answer.
3. **Temporal consistency**: when statements conflict, use the most recent valid statement supported by the conversation history.

Question and evidence. `Question: {question} Retrieved conversation evidence: {evidence}` *Choose the option that best fits this user's preferences and context.*

Options and answer format. question-specific options and required output format, followed by *Before finalizing, write exactly two short lines: (A) the most recent turn number in the evidence that states a preference relevant to the question, and what it says; (B) whether the user later asked to FORGET or CHANGE it. Then choose the option consistent with the most recent, non-forgotten preference and output 'Final Answer: [Letter]'.*

Output. two evidence lines followed by `Final Answer: [Letter].`

Figure 5: PersonaMem-V2 QA Prompt

A.3 Evaluation Prompt

For each benchmark, we use its official evaluation template to grade generated responses against the gold answers.

Grade one generated answer against the gold answer

Input. Question: {question} Gold answer: {answer} Model response: {response}

Prompt. *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.
```

Variants for the other question types.

- **Temporal reasoning**: standard criteria, with off-by-one day errors unpenalized.
- **Knowledge update**: the closing criteria change; earlier values may appear if the updated answer is given.
- **Preference**: judged against a desired-response rubric, which the answer need not reflect in full; the field is labeled `Rubric`.
- **Unanswerable**: the field becomes `Explanation`; a correct answer flags the question as unanswerable.

Output. `yes` or `no`, mapped to correct or wrong; any other output is an error.

Figure 6: Evaluation Prompt

B ALGORITHMS

This section presents the core algorithms of ReplayMem, including view generation and the complete retrieval process at answer time.

Algorithm 1: ReplayMem View Construction

Input: source chunks $\mathcal{C}$ and extraction prompt P
Output: key index K and chunk–key links L
for each source chunk $c \in \mathcal{C}$ **do**
  $out \leftarrow P(c, \text{same-session neighbours of } c)$
  if out is invalid, discard it and continue
  $z \leftarrow$ distinct terms in out
  for each view v and key $k \in z[v]$ **do**
    add or reuse (v, k) in K; add $(c, (v, k))$ to L
  end for
end for
return K, L

Algorithm 2: ReplayMem Retrieval

Input: query q, source chunks $\mathcal{C}$, key index K, chunk–key links L, and budget B
Output: trajectory T_q and rendered context X

$v \leftarrow \text{Embed}(q)$; $paths[\text{source}] \leftarrow \text{SearchChunks}(q, v, \mathcal{C})$
for $view \in \{\text{Content, State-Aspect}\}$ **do**
  $H_{view} \leftarrow \text{SearchKeys}(K, v, view)$
  $paths[view] \leftarrow \{c : (c, k) \in L \text{ for some } k \in H_{view}\}$
end for
$\mathcal{A}_q \leftarrow \text{Rank}(q, \text{Fuse}(paths))$ ▷ initial anchors
$\mathcal{E}_q \leftarrow \mathcal{A}_q$; $seen \leftarrow \text{ids}(\mathcal{A}_q)$; $frontier \leftarrow \mathcal{A}_q$
$Active \leftarrow \{k \in \cup_{view} H_{view} : k \text{ links to an anchor}\}$; $Used \leftarrow \emptyset$
while $Active \neq \emptyset$ **do**
  for key hit $h \in Active$ **do**
    $Used \leftarrow Used \cup \{h.\text{id}\}$
    $G_h \leftarrow \{h\}$ plus related unused keys of h's view
    $L_h \leftarrow \{c : (c, k) \in L \text{ for some } k \in G_h\} \setminus seen$
  end for
  $pool \leftarrow \text{Fuse}(\{L_h : h \in Active\})$; **if** $pool = \emptyset$ **then break**
  $frontier \leftarrow$ highest-ranked unseen chunks in $\text{Rank}(q, pool)$; add them to $\mathcal{E}_q$ and $seen$
  if $\text{Tokens}(\text{Render}(\mathcal{E}_q)) - \text{Tokens}(\text{Render}(\mathcal{A}_q)) \geq B$ **then break**
  $Active \leftarrow \{k : (c, k) \in L \text{ for some } c \in frontier\} \setminus Used$, ranked by similarity to v
end while
$T_q \leftarrow \text{Rank}(q, \mathcal{E}_q)$; $X \leftarrow \text{Render}(\text{Select}(T_q))$
return T_q, X

C ILLUSTRATIVE RETRIEVAL TRACE

This section presents a complete example of how ReplayMem uses semantic associations to successfully retrieve the evidence needed to answer a real LongMemEval question. This case illustrates the strong memory capabilities of the ReplayMem architecture.

C.1 EXAMPLE FOR RETRIEVAL

Figure 7 illustrates a retrieval trajectory from source chunks to the final answer. After initial retrieval over the source text, ReplayMem repeatedly expands retrieval through semantic associations to find the information needed to answer the question, ultimately enabling the model to produce the correct answer in the QA stage.

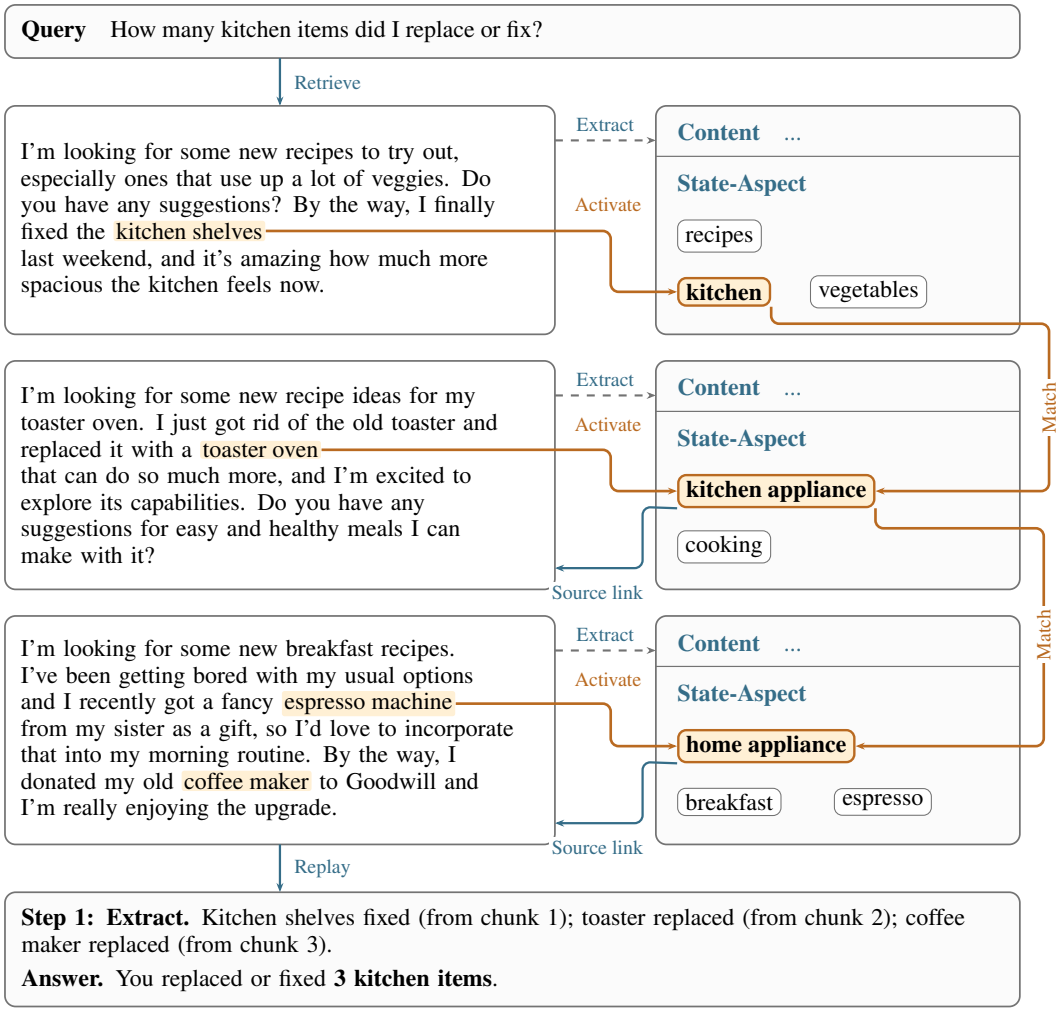

Figure 7: Illustrative key-guided retrieval trace. The query activates keys extracted from three source chunks; key matches connect the chunks across hops, and the final answer replays the linked source evidence.

C.2 Example Answer

Next, we show how ReplayMem correctly answers the question during the QA stage using the retrieved memories. The model progressively summarizes and reasons over the provided memories before producing the correct answer.

Example answer: counting the kitchen items that were replaced or fixed

Question. *How many kitchen items did I replace or fix?*

Step 1 — Evidence extracted from the retrieved source text.

- **Kitchen shelves** (fixed): "I finally fixed the kitchen shelves last weekend" [2023-05-20].
- **Toaster** (replaced): "I just got rid of the old toaster and replaced it with a toaster oven" [2023-05-26].
- **Coffee maker** (replaced): "I donated my old coffee maker to Goodwill and I'm really enjoying the upgrade" [2023-05-30].

Step 2 — Counting distinct items. The evidence mentions three distinct items that were replaced or fixed: the shelves (fixed); the toaster and the coffee maker (replaced). The espresso machine that appears alongside the coffee maker was received as a gift, and is not counted.

Step 3 — Answer. Three kitchen items were replaced or fixed: the shelves, the toaster, and the coffee maker.