TrajWiki: Source-Grounded Memory Trajectories for Long-Horizon Dialogue Agents

Jingyu Sun[1,2] **Yuyang Xue**[3] **Mingyang Li**[1] **Zhengtao Yao**[4]
Jiachen Li[5] Yang Cui[1] Wenhao Cai[1] Haozhe Liu[1]
Fangying Wang[1] Magdalene Katharina Montgomery[2]
Syed Murtuza Baker[1] Hongpeng Zhou[1,*]

[1]The University of Manchester [2]The University of Melbourne
[3]The University of Edinburgh [4]University of Southern California
[5]The University of Texas at Austin
[*]Corresponding author

Abstract

Large language model agents have shown strong capabilities in generating coherent and contextually appropriate responses, yet robust long-horizon dialogue remains limited by the lack of external memory that is traceable, updatable, and diagnostically transparent. Existing memory-augmented agents often store memories as isolated records or overwritable states, making it difficult to preserve how information originates, evolves, conflicts, or becomes obsolete over time. We propose **TrajWiki**, a trajectory-based memory framework for long-horizon conversational agents. Instead of treating memory as static entries, TrajWiki represents each memory as a source-grounded evolution trajectory, maintained through immutable episodic snapshots and claim-level operations such as `ADD`, `REVISE`, and `DEPRECATE`. To reduce fragmentation and retrieval cost, TrajWiki further introduces **Memory Wiki**, a persistent intermediate layer that incrementally compiles dialogue history into structured and interlinked wiki pages capturing salient entities, events, quantities, topics, and conflicts. At inference time, queries are routed hierarchically from relevant wiki pages to linked memory trajectories, then to corresponding snapshots and source messages for evidence-grounded answer synthesis. Experiments on LoCoMo and MedMT show that TrajWiki improves long-horizon dialogue performance across both open-source and closed-source LLM backbones, while providing greater interpretability and diagnostic visibility into memory evolution, retrieval failures, and answer generation.Code is available at `https://github.com/JingyuSunUOE/TrajWiki`.

1 Introduction

Large language model (LLM) agents are systems built around large language models that autonomously plan, reason, and act by iteratively interacting with users, external tools, and environments to solve complex, multi-step tasks, and they have shown early success in domains such as coding, research assistance, and broader task automation [1–8]. These systems rely on maintaining and updating context across successive interaction steps to support coherent decision-making; however, sustaining coherent long-horizon interaction remains difficult because the underlying models are constrained by finite context windows [9–11]. This limitation becomes particularly acute in long-term dialogue, where useful responses often depend on user-specific facts, evolving preferences, prior commitments, temporally distant events, and evidence scattered across multiple sessions [12–17].

Preprint.

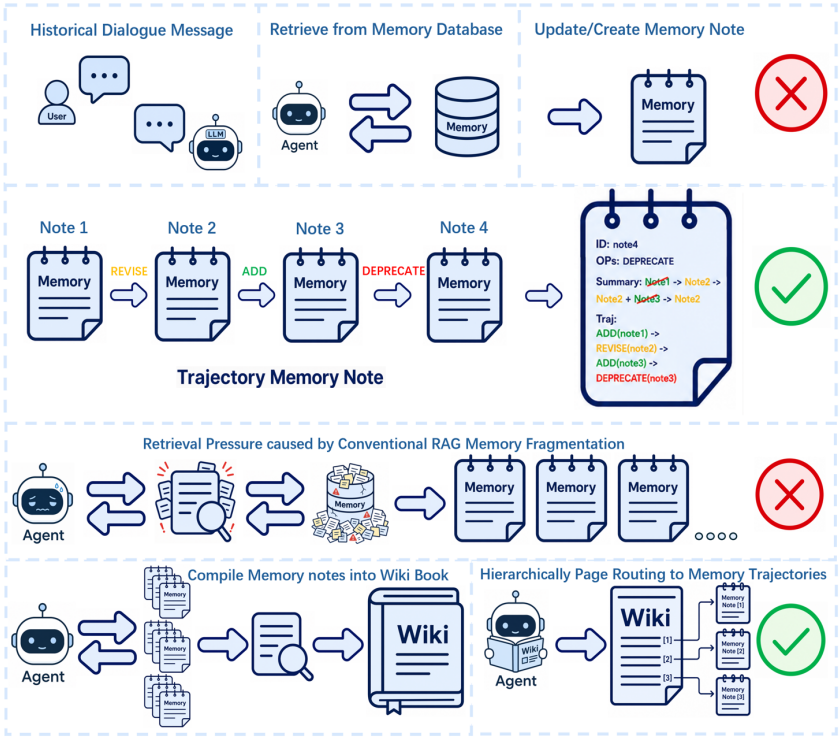

Figure 1: Comparison of conventional agent memory and TrajWiki's source-grounded trajectory memory and Memory Wiki. Conventional memory retains only the current belief and retrieves flat, isolated memories. TrajWiki preserves source-grounded update trajectories and structures them into a Memory Wiki, enabling reasoning over how facts evolve, conflict, and relate across long-horizon dialogue.

Simply extending the prompt with more history is costly and brittle, and does not provide reliable mechanisms for updating, organizing, or diagnosing long-term memory [18–22].

Recent work addresses this limitation by equipping LLM agents with explicit external memory. Most memory-augmented systems follow a write/read paradigm: dialogue history is compressed into memory units such as notes, facts, user profiles, or event records, and relevant units are later retrieved to support generation [23]. Other approaches improve memory organization through adaptive structures, dynamic linking, or graph-based retrieval [22]. While these methods demonstrate the value of persistent memory, we argue that two challenges remain underexplored, as shown in Figure 1. First, existing systems often retain only the current memory state without explicitly preserving its source-grounded update history. A retrieved memory may state what the agent currently believes, but not how that belief was formed, which claims were revised or deprecated, or which source utterances justify each update. Second, as fine-grained memories accumulate, flat retrieval over isolated memory items becomes increasingly fragmented, especially for temporal, multi-hop, or update-sensitive questions. Conventional retrieval-augmented generation treats knowledge mainly as searchable passages [24], whereas long-horizon dialogue memory also requires modeling how facts evolve, conflict, and relate over time.

In this paper, we propose **TrajWiki**, an interlinked, wiki-style, trajectory-based memory framework for long-horizon conversational agents. TrajWiki addresses the two challenges above through two key mechanisms. First, rather than representing memory as isolated and static records, it models each memory as a source-grounded evolutionary trajectory in which newly generated dialogue segments are stored as immutable episodic snapshots and linked to prior memory states through claim-level operations. Second, instead of repeatedly retrieving relevant information from fragmented raw chunks at query time, it reduces memory fragmentation by organizing trajectories into structured and interlinked wiki pages centered on entities, topics, events, and discrete factual units. At inference

time, TrajWiki performs hierarchical retrieval: a query is first routed to relevant wiki pages, then to associated memory trajectories, and finally to episodic snapshots and original source messages, enabling evidence-grounded response generation. In summary, our contributions are fourfold:

- We propose a source-grounded memory trajectory model that records long-term memory evolution through immutable episodic snapshots and claim-level update operations, preserving both the current effective memory and the historical path through which it was produced and making memory evolution inspectable at the level of claims and supporting evidence.
- We introduce an intermediate knowledge organization layer, **Memory Wiki**, that compiles trajectories into structured and interlinked wiki pages for more diagnosable retrieval, while preserving links to trajectories and source messages.
- We further introduce a hierarchical retrieval mechanism over interlinked memory trajectories that separates semantic organization, temporal memory evolution, and fine-grained evidence verification, reducing repeated reasoning over raw dialogue history while retaining provenance for diagnosis.
- We evaluate TrajWiki on LoCoMo and MedMT-Bench [16, 25], covering both open-domain very long-term conversation and domain-specific long multi-turn medical dialogue. Experiments with open-source and closed-source LLM backbones show that TrajWiki improves long-horizon dialogue performance over strong memory-augmented baselines and provides richer diagnostic visibility into memory evolution, retrieval failures, and answer generation.

2 Related Work

2.1 Agent Memory Systems

External memory has become a central mechanism for LLM-based agents that operate beyond a fixed context window. Existing systems have explored multiple ways to store, update, and retrieve long-term interaction history. MemoryBank maintains user-specific memories with time- and importance-aware reinforcement, while MemGPT and MemoryOS frame memory management as a systems problem with hierarchical storage and context control [19, 20, 26]. More recent work introduces richer memory structures, including hierarchical memory trees, temporal knowledge graphs, dynamically linked memory notes, and scalable conversational memory consolidation [21, 22, 27, 28]. In long-horizon dialogue, timeline-based memory, reflective memory management, and graph-structured associative retrieval further demonstrate the value of structured memory for personalization and cross-session response generation [29–31]. Despite these advances, most prior systems focus primarily on what to store and how to retrieve it, rather than explicitly modeling how memory states evolve. Even when memories are updated or linked, the update process is rarely represented as a provenance-aware sequence of immutable snapshots with claim-level edit semantics. This makes it difficult to inspect which claims were added, revised, or deprecated, and which source interactions justified each change.

2.2 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) grounds language generation in external non-parametric memory by retrieving relevant evidence and conditioning generation on the retrieved content [24]. While standard RAG often retrieves flat top-k passages from relatively static corpora, recent methods improve retrieval with additional structure and multi-stage reasoning. Iter-RetGen interleaves retrieval and generation to refine evidence iteratively; RAPTOR organizes documents into a tree of recursive summaries; and GraphRAG builds graph-based indices to support global reasoning and query-focused summarization over large corpora [32–34]. TrajWiki shares the broader insight that retrieval benefits from structure, but targets a different setting. Existing RAG systems are mainly designed to query external document collections, whereas long-horizon dialogue agents must maintain an internal memory that evolves through ongoing interaction. In this setting, relevant evidence depends not only on semantic similarity, but also on when a memory was created, how it was revised, which earlier claims were superseded, and which source messages support the current state.

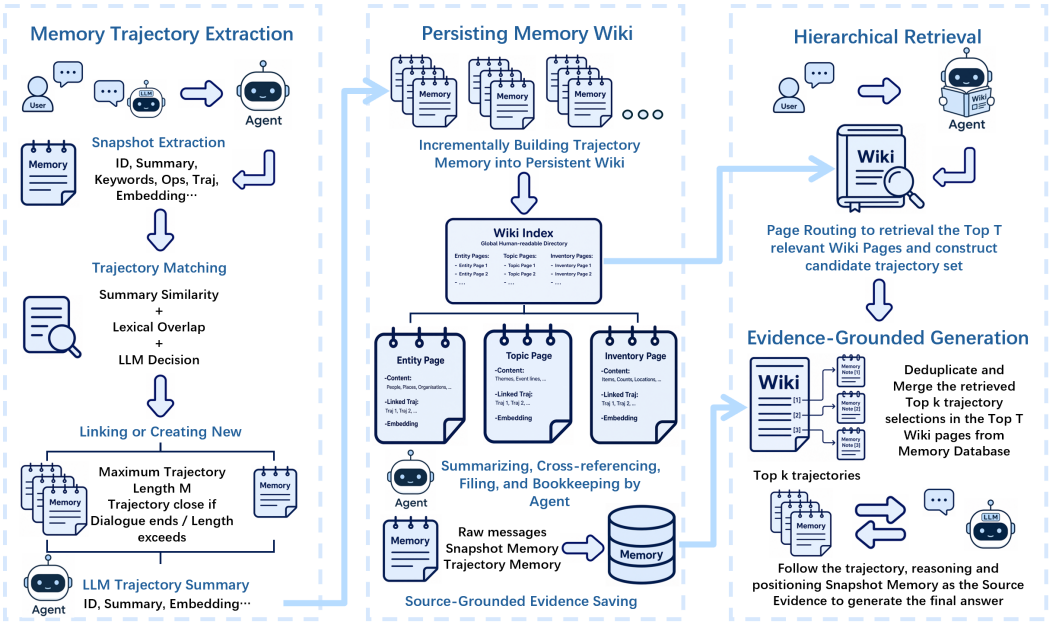

Figure 2: Overview of the TrajWiki architecture. TrajWiki converts dialogue history into source-grounded memory trajectories, compiles them into a persistent and interlinked Memory Wiki, retrieves evidence hierarchically from wiki pages to trajectories and source messages, and generates answers grounded in the retrieved evidence. The Memory Wiki serves as a compounding knowledge artifact, maintaining cross-references, flagged conflicts, and synthesized evidence as memory accumulates over time.

3 Methodology

We propose **TrajWiki**, as shown in Figure 2, a long-term memory framework for dialogue agents that represents memory as source-grounded trajectories and organizes them into a persistent Memory Wiki. Unlike standard retrieval-augmented generation, which retrieves external passages at inference time [24], TrajWiki first converts dialogue history into evolving memory trajectories, then compiles them into wiki-style pages for hierarchical retrieval and evidence-grounded answer generation.

Given a dialogue history

$$\mathcal{D} = \{m_1, \ldots, m_N\}, \qquad m_i = (x_i, u_i, \tau_i, \rho_i), \tag{1}$$

where x_i, u_i, τ_i, and ρ_i denote message content, speaker, timestamp, and source reference, respectively, TrajWiki follows the pipeline

$$\mathcal{D} \to \Sigma \to \mathcal{T} \to \mathcal{W} \to \mathcal{E}_q \to a_q. \tag{2}$$

Here, Σ denotes episodic snapshots, $\mathcal{T}$ memory trajectories, $\mathcal{W}$ the Memory Wiki, $\mathcal{E}_q$ the retrieved evidence for query q, and a_q the final answer. This design separates provenance tracking, persistent knowledge organization, and query-time grounding.

3.1 Provenance-Aware Memory Trajectories

TrajWiki stores memory as source-linked trajectories rather than isolated records. Each dialogue exchange is converted into an immutable episodic snapshot containing extracted claims, speaker and temporal context, and links to raw source messages. A memory trajectory is an ordered append-only sequence:

$$T_j = (\sigma_{j,1}, \sigma_{j,2}, \ldots, \sigma_{j,n_j}), \tag{3}$$

where each snapshot $\sigma_{j,t}$ remains traceable to the messages that produced it.

For a new snapshot σ, TrajWiki either appends it to the most compatible trajectory or initializes a new one:

$$T^\star = \arg\max_{T_j \in \mathcal{T}} g(\sigma, T_j), \qquad \sigma \mapsto \begin{cases} T^\star, & g(\sigma, T^\star) \geq \delta, \\ T_{\text{new}}, & \text{otherwise.} \end{cases} \tag{4}$$

The compatibility function g considers entity continuity, facet overlap, lexical evidence, temporal compatibility, and semantic drift, allowing related updates to be connected without collapsing unrelated facts into overly broad trajectories.

Within each trajectory, memory evolution is represented through claim-level edit operations:

$$C_{j,t} = \text{Apply}(C_{j,t-1}, \mathcal{O}_{j,t}), \qquad \mathcal{O}_{j,t} \subseteq \{\texttt{ADD}, \texttt{REVISE}, \texttt{DEPRECATE}\}. \tag{5}$$

Thus, revised or deprecated claims are not deleted; they remain linked to their original snapshots and source references, making memory updates inspectable over time.

3.2 Persistent Memory Wiki

Although trajectories preserve provenance, directly retrieving over all snapshots and claims can become fragmented as memory grows. TrajWiki therefore compiles trajectories into a persistent **Memory Wiki**, which serves as a readable and routable organization layer between raw memory and query-time retrieval.

Each wiki page is represented as

$$p = (\eta, y, h, L_p), \qquad L_p \subseteq \mathcal{T}, \tag{6}$$

where η is the page type, y is the title, h is the page content, and L_p is the set of linked trajectories. We use four page types:

$$\eta \in \{\texttt{index}, \texttt{entity}, \texttt{topic}, \texttt{inventory}\}.$$

The `index` page provides a global directory, `entity` pages organize recurring entities, `topic` pages capture events or thematic clusters, and `inventory` pages collect list-like or countable facts.

Wiki construction is summarized as

$$\mathcal{W} = \text{Compile}(\mathcal{T}; C^{\text{active}}, C^{\text{hist}}, A), \tag{7}$$

where C^{active} and C^{hist} denote active and historical claims, and A denotes source anchors and metadata. The wiki does not replace source evidence with summaries; instead, it organizes accumulated memory while preserving links to trajectories and raw messages. To avoid making trajectories reachable only through the global directory, TrajWiki encourages non-index coverage:

$$\bigcup_{p \in \mathcal{W}_{\text{non-index}}} L_p \approx \mathcal{T}. \tag{8}$$

3.3 Hierarchical Retrieval and Evidence-Grounded Generation

Given a query q, TrajWiki retrieves evidence through a hierarchy of wiki pages, trajectories, and source-linked snapshots. It first routes the query to relevant wiki pages and uses their links to construct a candidate trajectory set:

$$\mathcal{P}_q = \text{Top}_t\{S_p(q, p) : p \in \mathcal{W}\}, \qquad \mathcal{C}_q = \bigcup_{p \in \mathcal{P}_q} L_p. \tag{9}$$

Here, S_p combines semantic, lexical, entity, facet, and temporal signals. This stage narrows retrieval from the entire memory store to organized memory regions.

From $\mathcal{C}_q$, TrajWiki selects trajectories using a relevance-and-coverage objective:

$$\mathcal{R}_q = \text{Top}_k \left\{ b(q, T) + \alpha \Delta_{\text{cov}}(T) - \lambda \, \text{red}(T) : T \in \mathcal{C}_q \right\}. \tag{10}$$

The base score $b(q, T)$ measures relevance, Δ_{cov} rewards complementary entities, events, facets, or countable items, and red penalizes redundancy. This encourages retrieval of multiple complementary trajectories for list, count, temporal, and multi-hop questions.

Selected trajectories are expanded into provenance-linked evidence:

$$\mathcal{E}_q = \{(T, \sigma, c, \rho, \tau) : T \in \mathcal{R}_q,\ \sigma \in T,\ c \in C_T\}. \tag{11}$$

Thus, retrieved evidence contains not only text fragments, but also their trajectory, claim, source reference, and timestamp.

The final answer is generated under a source-support constraint:

$$z_q = \mathrm{Synth}_\theta(q, \mathcal{E}_q), \qquad a_q = \mathrm{Answer}(z_q), \qquad \mathrm{refs}(a_q) \subseteq \mathrm{refs}(\mathcal{E}_q). \tag{12}$$

The synthesis z_q records the answerability decision, supporting facts, source references, temporal anchors, and uncertainties. If the retrieved evidence is insufficient, TrajWiki abstains or performs a controlled retry by generating retrieval hints and rerouting through the wiki and trajectory layers. In all cases, the final answer must remain grounded in retrieved source evidence.

Overall, TrajWiki decomposes long-term dialogue memory into three layers: source-grounded trajectories for memory evolution, Memory Wiki pages for persistent organization, and hierarchical retrieval for query-time grounding. This enables agents to accumulate, revise, and query long-term memory while preserving the provenance needed to audit both memory updates and generated answers. We provide the reproducible implementation details, including deterministic scoring features, structured-output schemas, hyperparameters, validation rules, and retry/repair procedures, in Appendix A.

4 Experiment

4.1 Datasets and Evaluation

We evaluate **TrajWiki** on two long-horizon dialogue benchmarks. Our primary benchmark is LoCoMo, a long-term conversational memory dataset with gold question-answer pairs [16]. We use its QA task, which covers *single-hop*, *multi-hop* requiring cross-session evidence aggregation, *temporal reasoning*, and *open-domain / world-knowledge* questions. Following Mem0 [21], we exclude the adversarial category to ensure fair comparison with prior memory-augmented baselines. To assess robustness beyond open-domain conversations, we further use MedMT-Bench as a domain-specific stress test [25]. MedMT-Bench evaluates long multi-turn medical conversations using instance-level rubrics rather than a single reference answer. Since our goal is to evaluate memory mechanisms rather than medical knowledge or multimodal reasoning, we select memory-relevant cases from three categories: *Long-Context Memory and Understanding (LCMU)*, *Resistance to Contextual Interference (RCI)*, and the *Information Contradiction (IC)* subset of *Instruction Clarification*. These categories test long-history alignment, robustness to contextual noise, and conflict detection between historical facts and current user statements.

We compare TrajWiki with five baselines: **Full Context**, which directly conditions on the complete dialogue history; **Naive RAG**, evaluated with both session-level and turn-level retrieval granularity [24]; **LangMem**, which extracts salient conversational information for persistent memory retrieval [35]; **A-MEM**, which dynamically links structured memory notes [22]; and **Mem0**, which performs incremental memory extraction, consolidation, and retrieval [21]. For LoCoMo, we report F1 and BLEU-1 as lexical overlap metrics [36]. Since long-horizon dialogue answers may be semantically correct despite different surface forms, we additionally use a strict LLM-as-a-Judge protocol to evaluate answer correctness [37]. For MedMT-Bench, we follow its rubric-based setting and report test-point pass accuracy. Details of metric computation are provided in Appendix B.1.

4.2 Implementation Details

For all baselines, we follow their original configurations, hyperparameters, and system prompts. When details are underspecified, we use the closest publicly described setting while keeping the backbone model, answering format, and evaluation protocol consistent across methods. For closed-source experiments, we use `GPT-4o-mini` as the backbone model and LLM judge. For open-source experiments, we use `Qwen3-8B` and `Qwen3-32B` in non-thinking mode as backbone models, and `Qwen3-8B` as the judge. TrajWiki uses `Qwen3-Embedding-8B` for all embedding-based retrieval components. We report all major prompts in Appendix C. For TrajWiki, the default setting uses maximum trajectory length $m = 15$, routes each query through the top $t = 15$ wiki pages, and selects the top $k = 15$ trajectories. These settings balance retrieval coverage and computational cost.

Table 1: Overall performance comparison on LoCoMo and MedMT-Bench. For LoCoMo, we report F1, BLEU, and Acc separately for each question type. For MedMT-Bench, we report Acc for each subset. The best performance is marked in bold.

Model	Method	LoCoMo												MedMT-Bench		
		Multi Hop			Temporal			Open Domain			Single Hop			LCMU	RCI	IC
		F1	BLEU	Acc	F1	BLEU	Acc	F1	BLEU	Acc	F1	BLEU	Acc	Acc	Acc	Acc
GPT 4o-mini	Full Context	28.11	21.71	32.98	23.11	16.12	30.22	16.76	13.57	35.42	43.77	36.43	71.72	27.12	11.16	7.47
	Naive RAG	21.78	17.08	20.21	20.35	15.37	24.92	14.89	11.67	26.04	40.49	34.13	60.52	16.43	18.86	7.82
	LangMem	22.01	16.92	28.01	20.59	14.43	31.78	10.31	8.09	23.96	34.4	26.62	55.28	17.78	16.34	8.26
	A-MEM	27.02	20.09	37.58	41.52	35.6	26.17	12.44	9.97	23.96	38.74	34.03	50.54	12.51	17.12	9.33
	Mem0	34.52	25.13	40.07	**44.73**	37.86	30.56	13.12	9.5	25.02	37.19	33.64	44.48	17.11	14.52	12.17
	TrajWiki	**35.23**	**36.47**	**59.44**	42.42	**41.14**	**35.39**	**20.56**	**18.66**	**42.12**	**48.72**	**43.14**	**78.56**	**29.33**	**21.42**	**17.68**
Qwen3 32B	Full Context	27.79	20.91	49.65	32.47	26.56	32.27	15.52	11.44	39.67	29.53	32.71	**79.68**	29.63	13.16	9.42
	Naive RAG	24.22	19.01	35.82	22.61	18.21	20.25	12.88	8.7	36.46	35.19	27.27	69.08	19.63	21.05	7.62
	LangMem	26.01	24.37	38.16	24.9	23.81	20.28	9.15	6.87	22.92	36.61	34.88	44.15	18.18	16.67	7.74
	A-MEM	25.17	19.12	41.48	29.79	24.63	28.78	11.29	7.68	41.67	33.76	26.21	63.26	13.89	17.07	10.36
	Mem0	26.32	20.17	45.74	32.41	29.68	31.49	15.03	11.28	**42.7**	**42.58**	35.15	73.72	17.89	23.68	11.21
	TrajWiki	**31.42**	**30.56**	**50.33**	**33.48**	**31.26**	**37.33**	**17.86**	**16.44**	39.68	41.88	**40.42**	78.34	**33.42**	**27.33**	**15.34**
Qwen3 8B	Full Context	**39.05**	31.16	39.36	33.64	27.19	26.17	**20.67**	**16.52**	31.54	43.27	**43.45**	**78.21**	26.69	10.22	6.76
	Naive RAG	27.25	21.05	29.79	27.86	22.96	18.69	12.18	8.98	38.54	**46.32**	41.44	64.8	16.32	18.32	6.42
	LangMem	16.48	15.08	27.09	17.85	15.83	15.61	9.47	6.96	24.58	36.24	35.16	51.77	17.81	15.47	7.74
	A-MEM	27.54	22.08	35.46	**34.30**	28.38	21.18	11.39	8.62	22.29	42.3	37.66	59.45	12.04	16.32	8.76
	Mem0	31.73	24.82	38.27	28.96	26.24	20.16	15.03	11.28	21.18	42.58	35.15	57.64	15.42	13.62	10.52
	TrajWiki	30.25	**32.14**	**42.14**	31.48	**30.48**	**34.62**	16.89	13.52	**39.78**	43.46	37.24	63.48	**27.56**	**20.11**	**14.28**

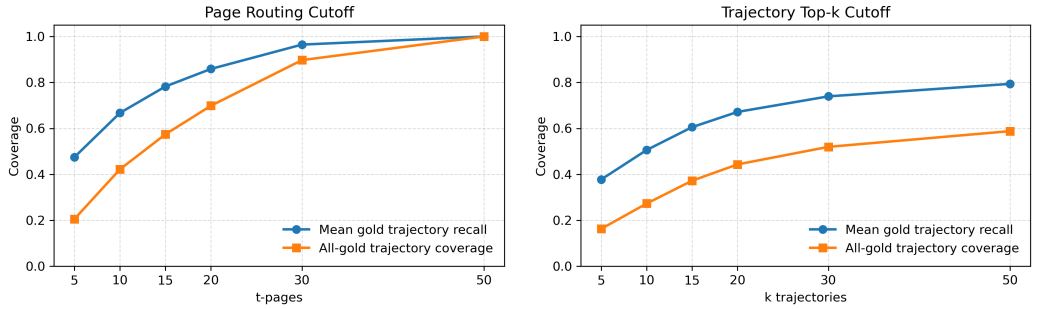

Figure 3: Hyperparameter Retrieval Cutoff Analysis on the LoCoMo. The left plot varies the wiki page routing cutoff t, showing how increasing routed pages improves gold trajectory coverage in the page-induced candidate universe. The right plot varies trajectory top-k, showing that selecting more trajectories improves recall but increases the evidence passed to answer generation.

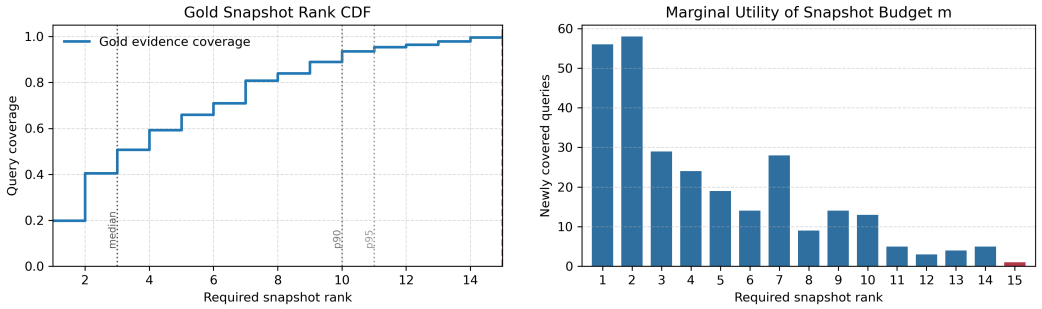

Figure 4: Snapshot Budget Analysis for trajectory length m on the LoCoMo. The left plot shows the cumulative fraction of queries whose gold evidence is covered within a given required snapshot rank. The right plot shows the marginal number of newly covered queries at each rank, indicating that most gains occur at shallow depths while $m = 15$ covers the remaining long-tail evidence.

Table 2: Counterfactual ablation on LoCoMO with GPT-4o-mini. All counterfactual variants use $k = 15$ and a 32K offline context budget without rerunning answer generation. "Unsup. Risk" is an unsupported-evidence risk proxy.

Setting	Removed / Restricted Component	Cand. Univ.	Gold Ref Cov.	Gold Traj. R@15	Ctx. Tokens	Unsup. Risk
Full TrajWiki	None	55.6	0.610	0.610	2.7K	0.645
Direct Trajectory Retrieval	Memory Wiki routing	130.4	0.356	0.346	25.6K	0.872
Wiki Summaries Only	Source-grounded trajectory expansion	41.2	0.000	0.807	25.3K	1.000
Flat Raw-Memory Retrieval	Trajectory and wiki organization	594.7	0.237	0.457	0.7K	0.918
Latest Snapshot Only	Historical snapshot depth	15.0	0.613	0.610	6.4K	0.649

4.3 Empirical Results

Table 1 reports the main results on LoCoMo and MedMT-Bench [16, 25]. Across both benchmarks and both GPT- and Qwen-based backbones, **TrajWiki** consistently outperforms most of the baselines, including Full Context, Naive RAG, LangMem, A-MEM, and Mem0 [21, 22, 35]. On LoCoMo, TrajWiki achieves better performance and improves over the strongest memory baseline, Mem0. The gains are most pronounced on multi-hop questions, where answering requires aggregating evidence distributed across sessions. This suggests that source-grounded trajectories and Memory Wiki routing help recover complementary memory traces beyond local semantic similarity. TrajWiki also shows a consistent advantage on temporal reasoning questions, although the margin is smaller, reflecting the difficulty of resolving temporally distant and evolving dialogue evidence. On MedMT-Bench, the performance gap becomes more substantial. Most baselines degrade in this domain-specific setting, especially on the Information Contradiction subset, where models must detect conflicts between historical facts and the current user statement. In contrast, TrajWiki remains robust across the three medical scenarios, demonstrating that its memory mechanism transfers beyond open-domain conversations to high-constraint professional dialogues. We attribute this improvement to two aspects of the design: trajectory-level memory preserves how claims evolve over time, while wiki-based routing reduces retrieval fragmentation before evidence-level expansion.

4.4 Hyperparameter Analysis

We analyze three key hyperparameters of TrajWiki on LoCoMo: the maximum trajectory length m, the number of routed wiki pages t, and the number of selected trajectories k. Rather than rerunning answer generation for every setting, we use the compact retrieval diagnostics saved during benchmarking to perform an offline coverage analysis. Figure 3 compares the effects of page routing cutoff and trajectory top-k. Increasing t substantially improves the probability that gold trajectories enter the page-induced candidate universe: small values such as $t = 5$ or $t = 10$ miss many supporting trajectories, while $t = 15$ provides a practical trade-off and $t = 20$ further improves coverage at the cost of a larger candidate set. Increasing k similarly improves gold trajectory coverage, but also increases the amount of evidence passed to answer generation, which may introduce additional noise. Figure 4 analyzes the required snapshot depth inside gold trajectories. Most queries require only shallow trajectory history, while a small long tail requires deeper snapshots. The marginal utility plot shows that most additional coverage is gained at small ranks, with diminishing returns after $m = 10$, while $m = 15$ covers the remaining long-tail evidence. These results support the design of TrajWiki as a hierarchical memory system: wiki routing controls retrieval breadth, trajectory selection controls evidence depth, and bounded trajectories preserve sufficient historical context without requiring unbounded memory expansion.

4.5 Counterfactual Ablation

We conduct a counterfactual ablation on the LoCoMo multi-hop split using GPT-4o-mini. The goal is to isolate how TrajWiki's memory organization affects evidence retrieval, grounding, and context efficiency. Since the counterfactual variants are derived from a single full TrajWiki run and do not rerun answer generation, we report retrieval and context-grounding metrics for the ablated variants; observed answer-level metrics are available only for the full TrajWiki run. A more detailed budget-controlled analysis is provided in Appendix B.2. Table 2 shows the main results at $k = 15$ under a 32K offline context budget. Removing the Memory Wiki and directly ranking trajectories increases the candidate universe from 55.6 to 130.4 trajectories, while reducing gold source-reference coverage from 0.610 to 0.356. This suggests that wiki routing provides useful semantic organization

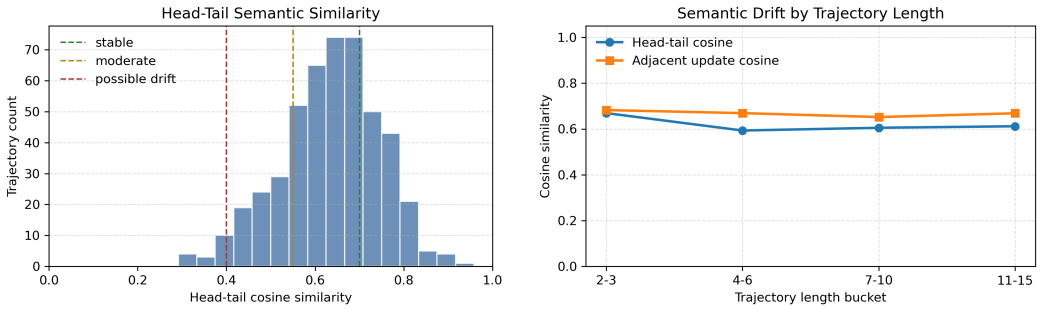

Figure 5: Trajectory Analysis. The left panel shows head-tail semantic similarity for non-singleton trajectories, while the right panel breaks semantic drift down by trajectory length and contrasts global head-tail span with local adjacent-snapshot similarity. Most trajectories remain semantically coherent, and longer trajectories mainly increase global span rather than causing unstable local updates.

rather than acting as a simple lossy filter. The wiki-only setting further shows that summaries alone are insufficient for source-grounded QA: although they can point to relevant trajectories, they do not recover gold source references. Flat raw-memory retrieval also has a much larger candidate universe and higher unsupported-evidence risk. Overall, the results support TrajWiki's combination of persistent wiki routing, source-grounded trajectories, and snapshot-level evidence expansion.

4.6 Trajectory Semantic Drift Analysis

We further analyze whether memory trajectories remain semantically coherent as they accumulate updates over time. For each trajectory, we compute the cosine similarity between the embeddings of its first and latest snapshots, which measures the semantic span of the trajectory, and the average cosine similarity between adjacent snapshots, which measures local continuity during memory updates. This analysis is performed offline using the saved snapshot embeddings from the LoCoMo run and does not involve additional model calls. As shown in Figure 5, the left panel indicates that most non-singleton trajectories maintain moderate to high head-tail similarity, while only a small fraction falls into the possible-drift region. The right panel further shows that longer trajectories naturally cover a broader semantic range, leading to lower head-tail similarity, but their adjacent-update similarity remains relatively stable. This suggests that TrajWiki trajectories support gradual memory evolution rather than arbitrary aggregation: even when a trajectory spans multiple related events, consecutive updates remain locally coherent. These results support the design of provenance-aware trajectories as a bounded yet extensible unit for long-term memory, where semantic drift can be measured and audited rather than hidden inside overwritten memory states.

4.7 Computational Cost and Scalability Analysis

We analyze the observed cost of TrajWiki on the LoCoMo using GPT-4o-mini. The cost is concentrated in memory construction and evidence organization rather than final answer generation: excluding benchmark-only evaluation, the run uses 43.28M deployment tokens, including 23.71M for reusable memory construction, 10.84M for query-time retrieval, and 6.92M for answer generation. This reflects TrajWiki's main trade-off: additional computation is used to build source-grounded trajectories and a persistent Memory Wiki. The scalability analysis further shows that the Memory Wiki reduces the query-time trajectory search space. Across dialogue samples, direct retrieval would consider 130.4 trajectories per query on average, while wiki routing reduces this to 55.6 candidate trajectories, a 2.35$\times$ reduction. Detailed phase-level costs, latency breakdowns, and memory-scaling results are provided in Appendix B.3.

4.8 Diagnostic Visibility and Limitations

TrajWiki preserves source links across snapshots, trajectories, Wiki pages, retrieved evidence, and final answers. We use *diagnostic provenance* to denote observable evidence paths and associated risk indicators. These records support inspection of memory evolution and answer construction, but

Table 3: **Exploratory human audit of 24 cases.** Packet tokens and source rows are means; verification and localization scores are percentages. The reference support and failure-stage labels were adjudicated by two developers. K denotes thousands of tokens.

Audit view	Packet tokens	Source rows	Support verification	Exact stage	Stage family
TrajWiki packet	5.93K	79.2	83.3	37.5	45.8
Full dialogue	26.74K	598.8	100.0	25.0	29.2

do not establish the correctness or completeness of either. The following analyses characterize the benefits of this visibility and the remaining limitations of the architecture.

Evidence retention and generation failures. TrajWiki depends on structured LLM outputs for extraction, memory updating, Wiki construction, answer synthesis, and validation. Malformed outputs can trigger expensive fallback and repair paths, particularly when models do not reliably follow structured-output instructions. Repeated summarization can also alter the salience of individual facts: retaining a source link does not ensure that the corresponding evidence remains prominent in the generated answer. In the original 282-query diagnostic run, the unsupported-answer-risk proxy was 0.610, while mean gold-reference coverage in final answer support was only 0.332. The former is an automated risk indicator, not a human-verified hallucination rate.

The diagnostic funnel on 200 queries further identifies where evidence is lost. Using each query's gold references as the denominator, mean retention was 100.0% in raw memory, 81.7% after Wiki reachability, 62.3% after trajectory selection, approximately 62% in the final source context, and 33.6% in final answer support. Evidence is therefore lost both before retrieval is completed and when retrieved information is converted into answer support. These measurements describe source-reference retention, not independently verified factual correctness. Appendix B.4 provides the underlying diagnostics and a case study of information loss through repeated summarization.

Exploratory human audit. We conducted a counterbalanced audit of 24 cases with two auditors who were not involved in system development. Each case was assessed under both the TrajWiki-packet and full-dialogue views, with different auditors assigned to the two views. Two developers adjudicated support and failure-stage labels before unblinding. Table 3 reports the results.

The full-dialogue view contained approximately $4.51\times$ as many tokens as the TrajWiki packet, while the packet yielded numerically higher exact-stage and stage-family localization scores. However, full dialogue provided better support verification, and exact-stage localization remained modest under both views. With only 24 cases and developer-adjudicated reference labels, this is preliminary evidence. The audit supports a more compact inspection surface with preliminary localization benefits.

5 Conclusion

We introduced **TrajWiki**, a long-term memory framework that represents dialogue history as provenance-aware memory trajectories and organizes them into a persistent Wiki-style semantic layer. By separating memory evolution and persistent organization, TrajWiki enables agents to accumulate, revise, and retrieve long-term conversational knowledge while preserving source-level evidence. Our experiments show that hierarchical retrieval over Wiki pages and trajectories improves the ability to locate multi-hop evidence, while evidence-grounded answer generation and post-hoc diagnostics make failures more interpretable. The results also highlight an important trade-off. TrajWiki incurs additional computational cost, but this cost supports reusable semantic organization, provenance tracking, and auditable retrieval, and can be amortized across future queries in long-term agent settings. Overall, TrajWiki demonstrates that treating memory as an evolving, source-grounded structure rather than a flat retrieval pool is a promising direction for building more reliable long-term dialogue agents.

References

[1] Sadia Sultana Chowa, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raiaan, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. From language to action: a review of large language models as autonomous agents and tool users. *Artificial Intelligence Review*, 2026.

[2] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[3] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

[4] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[5] Guozhi Yuan, Youfeng Liu, Jingli Yang, Wei Jia, Kai Lin, Yansong Gao, Shan He, Zilin Ding, and Haitao Li. Poact: Policy and action dual-control agent for generalized applications. *arXiv preprint arXiv:2501.07054*, 2025.

[6] Guangya Wan, Mingyang Ling, Xiaoqi Ren, Rujun Han, Sheng Li, and Zizhao Zhang. Compass: Enhancing agent long-horizon reasoning with evolving context. *arXiv preprint arXiv:2510.08790*, 2025.

[7] Mrinal Rawat, Ambuje Gupta, Rushil Goomer, Alessandro Di Bari, Neha Gupta, and Roberto Pieraccini. Pre-act: Multi-step planning and reasoning improves acting in llm agents. *arXiv preprint arXiv:2505.09970*, 2025.

[8] Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

[9] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 2978–2988, 2019.

[10] Aly Lidayan, Jakob Bjorner, Satvik Golechha, Kartik Goyal, and Alane Suhr. Abbel: Llm agents acting through belief bottlenecks expressed in language. *arXiv preprint arXiv:2512.20111*, 2025.

[11] Orit Shahnovsky and Rotem Dror. Ai planning framework for llm-based web agents. *arXiv preprint arXiv:2603.12710*, 2026.

[12] Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. Do llms recognize your preferences? evaluating personalized preference following in llms. *arXiv preprint arXiv:2502.09597*, 2025.

[13] Hideaki Joko, Shubham Chatterjee, Andrew Ramsay, Arjen P De Vries, Jeff Dalton, and Faegheh Hasibi. Doing personal laps: Llm-augmented dialogue construction for personalized multi-session conversational search. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 796–806, 2024.

[14] Xiaotian Zhang, Yuan Wang, Ruizhe Chen, Zeya Wang, Runchen Hou, and Zuozhu Liu. Towards proactive personalization through profile customization for individual users in dialogues. *arXiv preprint arXiv:2512.15302*, 2025.

[15] Qianyun Guo, Yibo Li, Yue Liu, and Bryan Hooi. Towards realistic personalization: Evaluating long-horizon preference following in personalized user-llm interactions. *arXiv preprint arXiv:2603.04191*, 2026.

[16] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, 2024.

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

[18] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36: 74530–74543, 2023.

[19] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 19724–19731, 2024.

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

[21] 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.

[22] 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.

[23] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

[24] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[25] Lin Yang, Yuancheng Yang, Xu Wang, Changkun Liu, and Haihua Yang. Medmt-bench: Can llms memorize and understand long multi-turn conversations in medical scenarios? *arXiv preprint arXiv:2603.23519*, 2026.

[26] 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.

[27] Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for llms. *arXiv preprint arXiv:2410.14052*, 2024.

[28] 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.

[29] Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seung-won Hwang, Dongha Lee, and Jinyoung Yeo. Towards lifelong dialogue agents via timeline-based memory management. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8631–8661, 2025.

[30] Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, et al. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8416–8439, 2025.

[31] Yujie Zhang, Weikang Yuan, and Zhuoren Jiang. Bridging intuitive associations and deliberate recall: Empowering llm personal assistant with graph-structured long-term memory. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17533–17547, 2025.

[32] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, 2023.

[33] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*, 2024.

[34] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[35] LangChain Team. LangMem: Long-term memory for ai agents. `https://langchain-ai.github.io/langmem/`, 2025. Software documentation. Accessed: 2026-05-06.

[36] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[37] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

APPENDIX

A Reproducible Implementation Details

This appendix provides the operational details needed to reproduce the main TrajWiki pipeline. TrajWiki combines deterministic scoring and validation with structured LLM calls. We therefore distinguish between deterministic candidate construction, structured model decisions, and fallback rules.

A.1 Memory construction.

Dialogue messages are first grouped into exchange-level inputs. For each exchange, TrajWiki asks the backbone model to produce an episodic memory seed with three fields: `summary_content`, `context`, and `keywords`. A separate claim extraction stage then emits source-grounded atomic claims:

$$c = (\text{text}, \text{status}, \text{source_message_ids}, \text{supporting_quote}),$$

where status $\in \{\texttt{active}, \texttt{deprecated}, \texttt{contradictory}, \texttt{needs-confirmation}\}$. All claims are validated against the source messages. Unsupported source ids, empty claims, and ungrounded exact terms are discarded. If the claim extractor fails or omits high-confidence source details, a deterministic preservation fallback extracts must-preserve surfaces such as names, places, books, instruments, activities, counts, temporal expressions, and quoted titles from the raw exchange.

A.2 Trajectory matching.

For each new episodic snapshot σ, TrajWiki first builds a deterministic feature vector from the snapshot text, extracted claims, and linked source messages. The feature set includes:

$$K_\sigma,\ E_\sigma,\ X_\sigma,\ F_\sigma,$$

where K_σ are lexical keywords, E_σ are entity mentions, X_σ are exact source-backed terms, and F_σ are relation/value facets such as `home_country`, `activity_location`, `research_topic`, and `event_type`. For each open trajectory T, TrajWiki stores analogous trajectory-level signals from active claims, historical item terms, retrieval summaries, and the latest snapshot.

Candidate trajectories are prefiltered by keyword/entity/facet overlap. If any overlap exists, only overlapping trajectories are kept, capped at 32 candidates by overlap count; otherwise all open trajectories are allowed as a fallback. Each candidate is scored by

$$\begin{aligned} g_{\text{det}}(\sigma, T) = {} & 0.60 \cos(e_\sigma, e_T^{\text{sum}}) + 0.20 \cos(e_\sigma, e_T^{\text{latest}}) + 0.20 J(K_\sigma, K_T) \\ & + b_{\text{entity}} + b_{\text{facet}} + b_{\text{cont}} - p_{\text{mismatch}}. \end{aligned}$$

Here J is keyword Jaccard overlap. The entity bonus is 0.08 when normalized entity keys overlap. The facet bonuses are 0.06 for exact facet-value overlap and 0.03 for facet-tag overlap. The continuity bonus is

$$b_{\text{cont}} = \min(0.14, 0.04|\text{specific}(\sigma) \cap \text{specific}(T)|).$$

A mismatch penalty of 0.10 is applied when a candidate shares only a broad entity but no specific continuity terms; a smaller penalty of 0.04 is applied when lexical overlap exists without specific continuity.

The top three candidates under g_{det} are then passed to a structured `trajectory_match` decision:

$$\texttt{decision} \in \{\texttt{CONTINUE}, \texttt{NEW}\}, \qquad \texttt{selected_candidate} \in \{T1, T2, T3, \texttt{null}\}.$$

If structured output is unavailable, TrajWiki falls back to a text parser. In mock deterministic mode only, the best candidate is continued when $g_{\text{det}} \geq \delta$, with $\delta = 0.72$. Thus, δ is a deterministic fallback threshold rather than the sole decision rule in remote-model experiments.

A.3 Claim evolution.

Within a trajectory, TrajWiki maintains an append-only claim lifecycle. Exact duplicate claims are first matched by normalized text. If the same claim text changes status, TrajWiki emits a status

operation: DEPRECATE when the new status is deprecated, otherwise REVISE. For unmatched new claims, previous claims are shortlisted by keyword overlap, keeping at most three candidate previous claims. A structured claim_transition_judge call then decides

$$\texttt{decision} \in \{\texttt{REVISE}, \texttt{ADD}\}.$$

A REVISE operation creates a new claim and marks the selected previous claim as deprecated; an ADD operation appends the claim without replacing prior evidence. If the judge fails or no previous claim candidate exists, TrajWiki falls back to ADD. Model-suggested operations are treated as hints only; the persisted state is derived by the validated deterministic procedure above.

A.4 Memory Wiki compilation.

After replaying a sample's dialogue into trajectories, TrajWiki compiles a sample-level Memory Wiki from the current trajectory store. In the implementation used for our experiments, wiki pages are refreshed after memory replay rather than edited one page at a time online. This preserves stable trajectory, snapshot, claim, and source ids while allowing page organization to be recomputed from the latest memory state.

The compiler first creates candidate page seeds:

$$\eta \in \{\texttt{index}, \texttt{entity}, \texttt{topic}, \texttt{inventory}\}.$$

The index page links all trajectories. entity pages group trajectories sharing an entity. inventory pages are created for list-like trajectories with multiple exact terms, display items, counts, or inventory-like facets. topic pages group repeated retrieval keywords. Broad entity pages are split into entity-facet pages when they exceed a trajectory threshold. Seeds with more than six trajectories are split into medium-granularity shards using trajectory-summary embeddings; the target page size is four trajectories and the maximum non-index page size is six. Redundant topic seeds are suppressed when at least 80% of their trajectories are already covered by entity or inventory seeds. A coverage audit then adds rescue pages for any trajectory reachable only through the index page.

Each page is compiled into markdown with required sections: overview, key facts, items/counts, linked trajectories, and conflicts/uncertainty. If the LLM page compiler returns empty text, missing sections, or placeholder-only content, TrajWiki falls back to deterministic markdown generated from linked trajectory summaries and evidence cards. Each page also stores a routing text, keywords, linked trajectory ids, page type, source-backed terms, display items, counts, and a page embedding.

A.5 Hierarchical retrieval.

Given a query q, TrajWiki first classifies the query shape using deterministic lexical rules: list-like, count-like, multi-entity, comparison-like, duration-count, and item family. It then embeds the query and routes it to wiki pages. Index pages are suppressed when non-index pages exist, so the wiki directory does not dominate retrieval.

Each page receives dense and sparse scores. The dense score combines page embedding similarity with entity, reflection, granularity, and answer-family adjustments:

$$s_p^{dense} = \cos(e_q, e_p) + b_{\text{entity}} + b_{\text{reflect}} + a_{\text{gran}} + 0.10 s_{\text{family}} - 0.12 p_{\text{family}}.$$

The sparse score is lexical overlap over page keywords, title, entity keys, exact terms, source-event terms, and family terms:

$$s_p^{sparse} = J(K_q, K_p) + 0.35 s_{\text{family}} - 0.25 p_{\text{family}}.$$

Dense and sparse rankings are fused by reciprocal rank fusion:

$$\text{RRF}(x) = \frac{1}{60 + r_{\text{dense}}(x)} + \frac{1}{60 + r_{\text{sparse}}(x)}.$$

The top page candidate pool is then optionally reranked by the backbone model with a structured textual output. If reranking fails, the RRF order is used.

Selected wiki pages induce a candidate trajectory universe:

$$\mathcal{C}_q = \bigcup_{p \in \mathcal{P}_q} L_p.$$

Candidate trajectories are scored similarly:

$$\begin{aligned} s_T^{dense} &= 0.75\cos(e_q, e_T^{sum}) + 0.15\cos(e_q, e_T^{latest}) + 0.10 b_{\text{entity/facet}} \\ &\quad + 0.08 s_{\text{family}} + 0.10 s_{\text{event}} - 0.04 p_{\text{family}}, \\ s_T^{sparse} &= J(K_q, K_T) + 0.60 s_{\text{family}} \\ &\quad + 0.75 s_{\text{event}} - 0.25 p_{\text{family}}. \end{aligned}$$

Dense and sparse trajectory ranks are again fused with RRF. The top trajectory pool is reranked by the backbone model when available.

For list, count, comparison, and multi-entity questions, TrajWiki performs coverage-aware greedy selection after reranking. The first selected trajectory is the strongest relevance anchor. Subsequent selections maximize

$$\begin{aligned} S(T) = s_{\text{rank}}(T) &+ 0.90\Delta_{\text{entity}} + 0.65\Delta_{\text{facet}} + 0.30\Delta_{\text{item}} + 0.12\Delta_{\text{term}} \\ &+ 0.18\mathbb{I}_{inventory} + 0.18\mathbb{I}_{count} + 0.75 s_{\text{family}} - p_{\text{red}}, \end{aligned}$$

where p_{red} penalizes repeated entities, repeated terms, repeated item terms, repeated clusters, and answer-family mismatch. For non-coverage queries, TrajWiki simply fills from the reranked order.

Evidence expansion and final answer. Selected trajectories are expanded into snapshots. TrajWiki first keeps the top snapshot per selected trajectory, then fills the remaining snapshot budget by query-snapshot embedding similarity. The snapshot budget is $2k$. Depending on the expansion mode, neighboring snapshots and update-linked snapshots are also added. Source messages linked to selected snapshots are compacted into a final prompt context containing wiki pages, selected trajectory summaries, snapshots, active claims, source messages, temporal anchors, and conflict/deprecated-claim diagnostics.

LoCoMo answer generation uses a structured evidence synthesis schema:

`can_answer`, `answer_type`,
`final_answer`, `supporting_facts`,
`supporting_source_refs`,
`counted_events`, `excluded_events`,
`uncertainties`, `abstain_reason`.

All supporting refs must be visible source refs in the retrieved context. Invalid refs are removed. If an answer has no valid source support after validation, TrajWiki converts it to an abstention. Count answers are additionally validated by removing duplicate, future/planned, uncertain, or unsupported events. Type mismatches trigger a typed retry or safe abstention.

A.6 Retry and repair.

Retrieval reflection is used only for LoCoMo when the initial answer abstains or retrieval is weak. A retrieval bundle is considered weak if it has no source messages, no candidate trajectories, no selected snapshots, or no active claims in the answer context. Reflection produces a structured query rewrite with target entities, event terms, temporal terms, must-find terms, candidate wiki slugs, and raw search terms. TrajWiki reroutes through the wiki using these hints. If the reflected answer still abstains or the synthesis schema returns `can_answer=false`, a raw-message rescue pass retrieves lexical/semantic raw source candidates without changing the memory state.

Post-generation repair is triggered by deterministic checks for unsupported extra list items, unsupported counts, missing supported list items, over-generic items, scope-mismatched items, unresolved bridge aliases, or answer-type mismatch. Repairs are generated using only retrieved evidence. A repaired answer is discarded if it is malformed, loses required supported values, introduces unsupported extras, or fails arbitration; in such cases the initial answer is kept or TrajWiki emits a safe abstention.

A.7 Hyperparameters.

Unless otherwise stated, the LoCoMo experiments use $m = 15$ memory trajectories/pages exposed to downstream stages, $t_{\text{pages}} = 15$ wiki pages, $k = 15$ selected trajectories, neighbor radius 1, retrieval

expansion mode `update_linked_plus_neighbors`, trajectory candidate prefilter cap 32, rerank candidate-pool size 12, RRF constant 60, maximum non-index wiki page size 6, target wiki page size 4, and mock-mode trajectory continuation threshold $\delta = 0.72$. The implementation supports remote, local, and OpenAI-compatible providers; structured-output schemas are used when supported, with parser and deterministic fallbacks otherwise.

B Evaluation Details

B.1 Evaluation Metrics

We report two automatic text-based metrics, F1 and BLEU-1, following common practice in long-form question answering benchmarks. In our implementation, both metrics are computed over a canonicalized semantic representation rather than raw surface text. This reduces spurious penalties when an answer expresses the correct fact using longer or slightly different natural-language phrasing.

Given a reference answer y and a model answer $\hat{y}$, we first extract canonical answer slots from both answers. Let

$$\mathcal{C}(y) = \{(s_i, v_i)\}_{i=1}^{n}, \qquad \mathcal{C}(\hat{y}) = \{(\hat{s}_j, \hat{v}_j)\}_{j=1}^{m},$$

where s denotes a semantic slot identifier and v denotes a normalized canonical value.

F1. We compute F1 using slot-aware semantic matching between reference and candidate canonical values. Unlike exact set overlap, our current implementation uses a conservative soft matching policy within the same slot. A candidate value may match a reference value if they are identical after normalization, differ only by articles, punctuation, or simple inflection, contain benign modifiers such as "LGBTQ support group" versus "support group", or match through a predefined alias relation such as "NYC" and "New York City". Contrastive modifiers, numbers, colors, person names, and other distinguishing terms are preserved to avoid over-matching.

Formally, let

$$\mu\big((\hat{s}, \hat{v}), (s, v)\big) \in \{0, 1\}$$

denote whether a candidate slot-value pair matches a reference slot-value pair under this policy. Matching is restricted to pairs with $\hat{s} = s$, and each candidate value can match at most one reference value. We therefore compute a maximum one-to-one matching

$$M^\star = \arg\max_{M} \sum_{((\hat{s},\hat{v}),(s,v))\in M} \mu\big((\hat{s}, \hat{v}), (s, v)\big).$$

Precision, recall, and F1 are then

$$P = \frac{|M^\star|}{|\mathcal{C}(\hat{y})|}, \qquad R = \frac{|M^\star|}{|\mathcal{C}(y)|},$$

$$\mathrm{F1} = \frac{2PR}{P + R}.$$

If both canonical sets are empty, F1 is defined as 1.0; if only one side is empty, F1 is 0.0. This metric rewards complete semantic coverage of the reference facts, penalizes missing reference facts through recall, and penalizes unsupported extra canonical facts through precision.

BLEU-1. We also report BLEU-1 as clipped unigram precision over canonical answer text. We linearize the canonical slot-value representation into a deterministic text sequence, tokenize it after lowercasing and punctuation normalization, and compute

$$\text{BLEU-1} = \frac{\sum_{w \in V} \min\left(\mathrm{count}_{\hat{y}}(w), \mathrm{count}_{y}(w)\right)}{\sum_{w \in V} \mathrm{count}_{\hat{y}}(w)}.$$

Unlike standard BLEU, we do not apply a brevity penalty. BLEU-1 therefore measures how much of the candidate's canonical unigram content is supported by the reference answer. If both token sequences are empty, BLEU-1 is defined as 1.0; if only one is empty, it is 0.0.

Together, F1 evaluates slot-level semantic fact coverage with conservative soft matching, while BLEU-1 measures canonical unigram precision. These deterministic metrics complement the LLM-based judge by providing reproducible lexical-semantic scores over normalized answer content.

Table 4: Full offline ablation at 32K context budget. All settings use $k = 15$. Counterfactual variants report retrieval/context proxies only.

Setting	Cand. Univ.	Gold Ref Cov.	Gold Traj. R@15	All Ref Rate	Ctx. Tokens	Unsup. Risk
Full TrajWiki	55.6	0.610	0.610	0.355	2.7K	0.645
Direct Trajectory Retrieval	130.4	0.356	0.346	0.128	25.6K	0.872
Wiki Summaries Only	41.2	0.000	0.807	0.000	25.3K	1.000
Flat Raw-Memory Retrieval	594.7	0.237	0.457	0.082	0.7K	0.918
Latest Snapshot Only	15.0	0.613	0.610	0.351	6.4K	0.649
Latest Two Snapshots	22.5	0.564	0.558	0.287	6.8K	0.713
Source-Linked Claims Only	347.0	0.088	0.136	0.011	0.2K	0.989

B.2 Offline Counterfactual Retrieval and Context Ablation

We provide a more detailed offline ablation analysis to complement the compact results in Table 2. The analysis is conducted on the LoCoMo multi-hop split using the same full TrajWiki run with GPT-4o-mini. The counterfactual settings reuse saved retrieval diagnostics, gold evidence labels, trajectory/page rankings, source references, and token estimates. They therefore evaluate evidence availability and context cost, but do not rerun answer generation. Consequently, we do not report true judge accuracy for counterfactual variants.

Metrics. **Candidate universe** is the average number of memory items considered before final selection. **Gold Ref Cov.** measures the fraction of gold source references recovered by the selected context. **Gold Traj. R@15** measures recall of gold trajectories under the top-15 trajectory cutoff. **All Ref Rate** is the fraction of queries for which all gold source references are recovered. **Ctx. Tokens** is an offline whitespace-based context token estimate. **Unsup. Risk** is a proxy for unsupported evidence risk, indicating that selected evidence lacks sufficient source support for the gold references.

Ablation settings. **Full TrajWiki** uses the observed wiki-routed retrieval and source-grounded trajectory expansion. **Direct Trajectory Retrieval** removes wiki routing and ranks all trajectories in the same dialogue sample. **Wiki Summaries Only** keeps wiki pages but does not expand them into source-linked trajectory evidence. **Flat Raw-Memory Retrieval** ranks raw dialogue messages directly using lexical signals. **Latest Snapshot Only** and **Latest Two Snapshots** restrict each selected trajectory to its most recent one or two snapshots. **Source-Linked Claims Only** keeps only claims with explicit source links and is included as a diagnostic proxy rather than a full source-support-constraint ablation.

Table 4 reports all variants at the 32K budget. The full TrajWiki setting provides the best balance between gold source-reference coverage, trajectory recall, and candidate-space size. Direct trajectory retrieval considers more than twice as many candidates but recovers less gold evidence under the same offline selection protocol. Wiki-only retrieval has high trajectory recall but zero gold source-reference coverage, showing that wiki summaries alone are not sufficient as answer evidence. Flat raw-memory retrieval is cheap under this lexical selector but has a very large memory universe and high unsupported-evidence risk. The source-linked-claims-only diagnostic has very low context cost but poor coverage, indicating that source filtering alone cannot replace structured retrieval.

Table 5 reports the budget-controlled results for the main settings. Full TrajWiki reaches its evidence coverage with a compact selected context. In contrast, direct trajectory retrieval improves as the budget increases, but even at 32K it remains below full TrajWiki in gold source-reference coverage and trajectory recall. Wiki-only retrieval increases trajectory recall with larger budgets but continues to have zero gold source-reference coverage, because wiki summaries are not source evidence. This supports the need for both hierarchical routing and source-grounded trajectory expansion.

B.3 Computational Cost and Scalability Analysis

We provide a detailed cost and scalability analysis for TrajWiki on the LoCoMo multi-hop split using GPT-4o-mini, with $m = 15$, $t_{\text{pages}} = 15$, and $k = 15$. Since TrajWiki uses multiple asynchronous workers, we report accumulated provider-side latency rather than end-to-end wall-clock time. We also separate deployment costs from benchmark-only evaluation costs. This section reports observed

Table 5: Budget-controlled offline ablation. Values are shown for $k = 15$. "Ctx." is estimated context tokens.

Setting / Budget	Gold Ref Cov.	Gold Traj. R@15	Ctx.	Unsup. Risk
Full TrajWiki, 4K	0.606	0.605	2.6K	0.649
Full TrajWiki, 8K	0.610	0.610	2.7K	0.645
Full TrajWiki, 16K	0.610	0.610	2.7K	0.645
Full TrajWiki, 32K	0.610	0.610	2.7K	0.645
Direct Trajectory Retrieval, 4K	0.041	0.041	0.6K	0.993
Direct Trajectory Retrieval, 8K	0.110	0.108	3.2K	0.979
Direct Trajectory Retrieval, 16K	0.216	0.210	9.9K	0.936
Direct Trajectory Retrieval, 32K	0.356	0.346	25.6K	0.872
Wiki Summaries Only, 4K	0.000	0.276	3.3K	1.000
Wiki Summaries Only, 8K	0.000	0.442	7.1K	1.000
Wiki Summaries Only, 16K	0.000	0.656	15.1K	1.000
Wiki Summaries Only, 32K	0.000	0.807	25.3K	1.000
Flat Raw-Memory Retrieval, 4K	0.237	0.457	0.7K	0.918
Flat Raw-Memory Retrieval, 8K	0.237	0.457	0.7K	0.918
Flat Raw-Memory Retrieval, 16K	0.237	0.457	0.7K	0.918
Flat Raw-Memory Retrieval, 32K	0.237	0.457	0.7K	0.918

Table 6: Phase-level cost breakdown for TrajWiki on LoCoMo multi-hop. Tokens and latency are aggregated across the full run. Benchmark evaluation is reported separately from deployment cost.

Phase	Deployment Scope	Reuse Scope	Calls	Tokens	Provider Latency
Memory construction	Deployment	Upfront reusable	2256	23.71M	25.66h
Query-time retrieval	Deployment	Per query	846	10.84M	1.58h
Answer generation	Deployment	Per query	564	6.92M	0.44h
Repair / validation	Deployment	Mixed	768	1.81M	2.50h
Benchmark evaluation	Benchmark only	Benchmark only	846	0.42M	0.45h

TrajWiki costs only; we do not claim a token break-even against proxy baselines without answer-level reruns.

Table 6 gives the phase-level cost breakdown. The largest source of token usage is memory construction, which consumes 23.71M tokens, followed by query-time retrieval with 10.84M tokens and answer generation with 6.92M tokens. This confirms that the dominant cost of TrajWiki is not the final answer itself, but the construction and organization of source-grounded long-term memory.

Figure 6 visualizes the same token distribution. Memory construction accounts for 54.3% of all measured tokens, query-time retrieval accounts for 24.8%, answer generation accounts for 15.8%, repair/validation accounts for 4.1%, and benchmark-only evaluation accounts for only 1.0%. Thus, evaluation overhead is not the main driver of the reported cost. Instead, the cost reflects TrajWiki's design choice to transform raw dialogue into trajectories, claims, snapshots, and wiki pages before answer generation.

Figure 7 shows an even stronger concentration in provider-side latency. Memory construction accounts for 83.8% of accumulated provider latency, compared with 8.1% for repair/validation, 5.2% for query-time retrieval, 1.5% for benchmark evaluation, and 1.4% for answer generation. The latency distribution indicates that memory-building operations such as claim processing, trajectory organization, and wiki construction are the main runtime bottlenecks. This suggests that future efficiency improvements should prioritize caching, batching, and replacing some LLM-based memory-construction steps with cheaper deterministic or distilled components.

Table 7 summarizes how memory size grows across dialogue samples. On average, each dialogue contains 588.2 raw messages, which TrajWiki organizes into 128.6 trajectories, 294.3 snapshots,

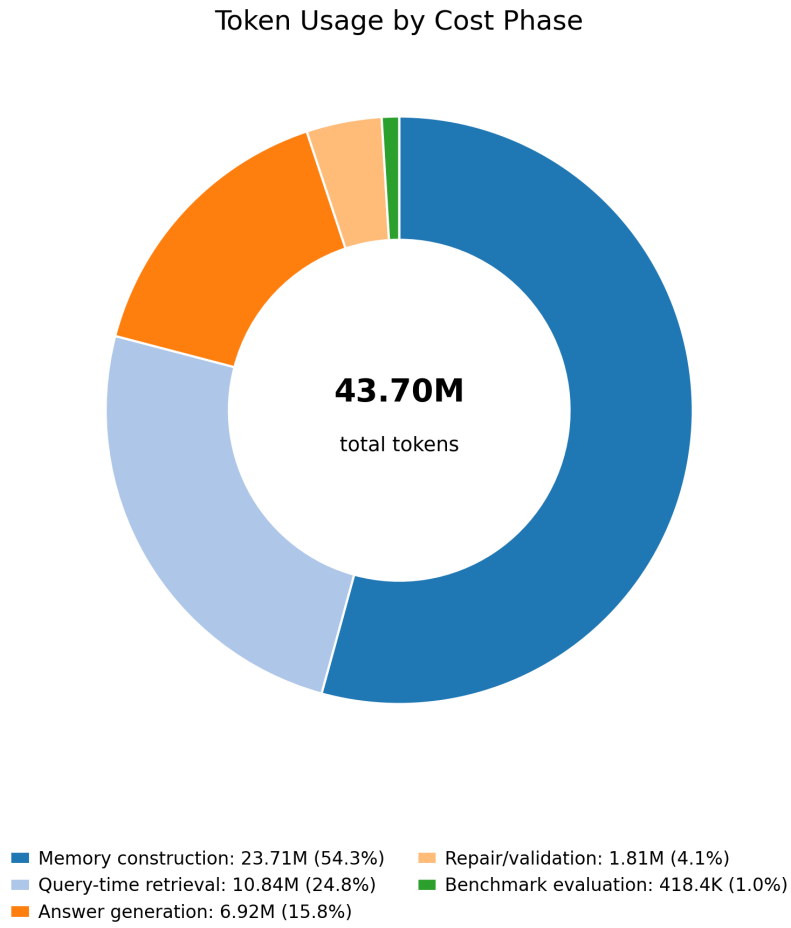

Figure 6: Token usage by cost phase. Token usage is dominated by memory construction and query-time retrieval, while benchmark-only evaluation contributes less than 1% of total measured tokens.

3409.2 claims, and 42.5 wiki pages. Figure 8 provides the same view across individual dialogue samples on a log scale. The figure shows that claims are the largest memory object class, while wiki pages remain much smaller in number than trajectories and snapshots. This supports the role of the Memory Wiki as a compact routing layer over a much larger provenance-rich memory store.

Figure 9 shows the query-time effect of this organization. Direct trajectory retrieval would consider 130.4 trajectories per query on average, whereas wiki routing reduces the candidate universe to 55.6 trajectories, a 2.35$\times$ reduction before final top-k selection. This reduction is important because query-time retrieval otherwise scales with the full trajectory store. The wiki layer therefore provides a concrete scalability benefit even though the current implementation remains token-intensive.

Overall, this analysis shows that TrajWiki is currently a computation-heavy, provenance-oriented memory system rather than a token-minimizing retrieval baseline. Its main efficiency benefit in this implementation is search-space reduction through persistent wiki routing. Future cost optimization should focus on reducing memory-construction latency, caching reusable retrieval computations, and replacing some repeated LLM-based organization steps with cheaper deterministic or distilled alternatives.

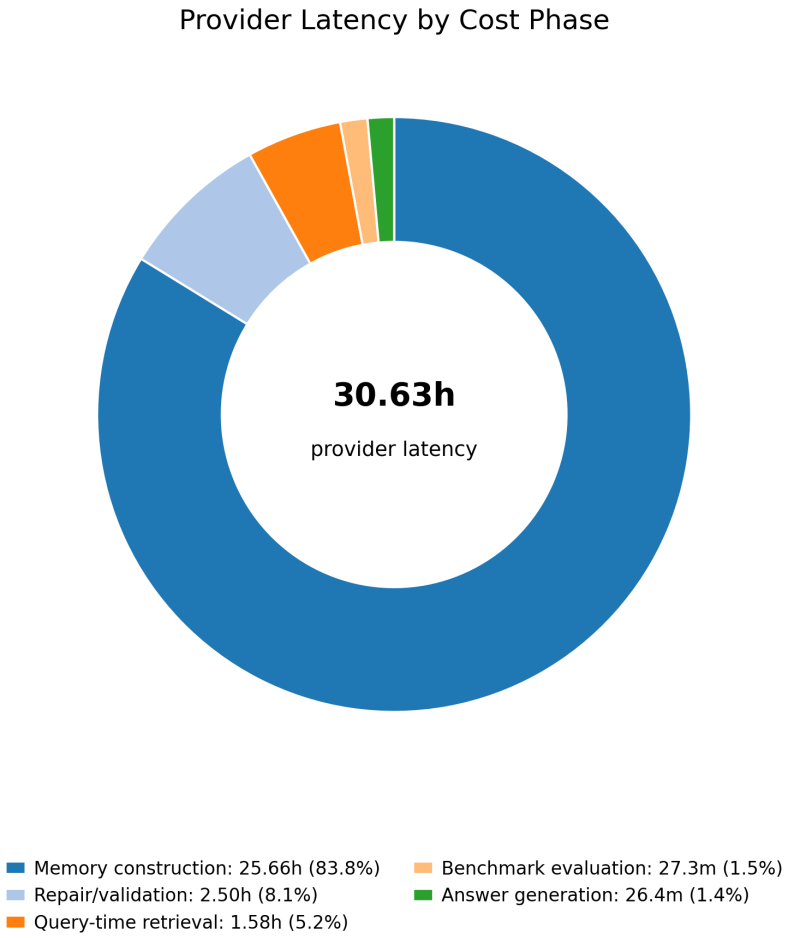

Figure 7: Provider-side latency by cost phase. Accumulated provider latency is primarily spent on memory construction, including extraction, trajectory organization, claim processing, and wiki compilation.

Table 7: Memory size and candidate scaling. Values are averaged across LoCoMo dialogue samples or evaluated queries.

Quantity	Mean	Min	Max
Raw messages per sample	588.2	369	689
Raw memory tokens per sample	19.9K	12.0K	23.9K
Trajectories per sample	128.6	43	188
Snapshots per sample	294.3	185	344
Claims per sample	3409.2	2613	4483
Wiki pages per sample	42.5	30	59
Direct trajectory universe per query	130.4	43	188
Wiki-routed trajectory universe per query	55.6	27	66
Selected trajectories per query	15.0	15	15
Selected snapshots per query	28.3	16	30
Selected source messages per query	74.2	29	150

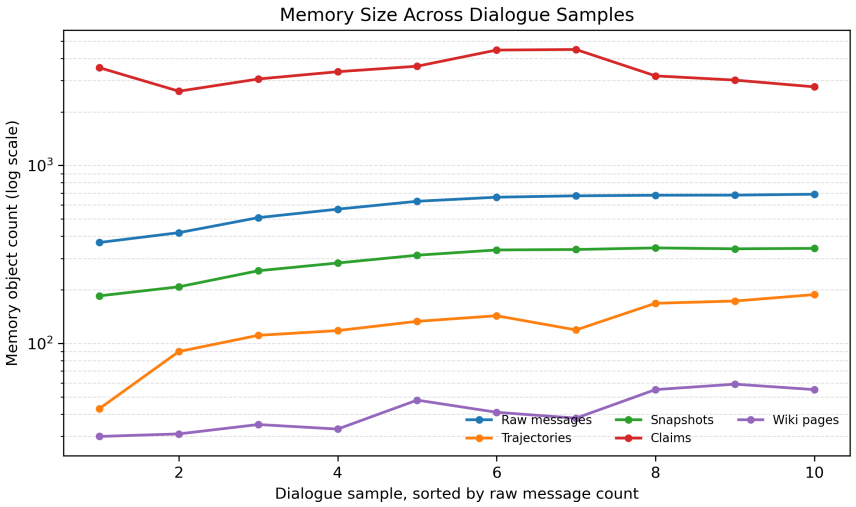

Figure 8: Memory size across dialogue samples. Counts are shown on a log scale. TrajWiki stores raw messages, trajectories, snapshots, claims, and wiki pages as separate memory objects.

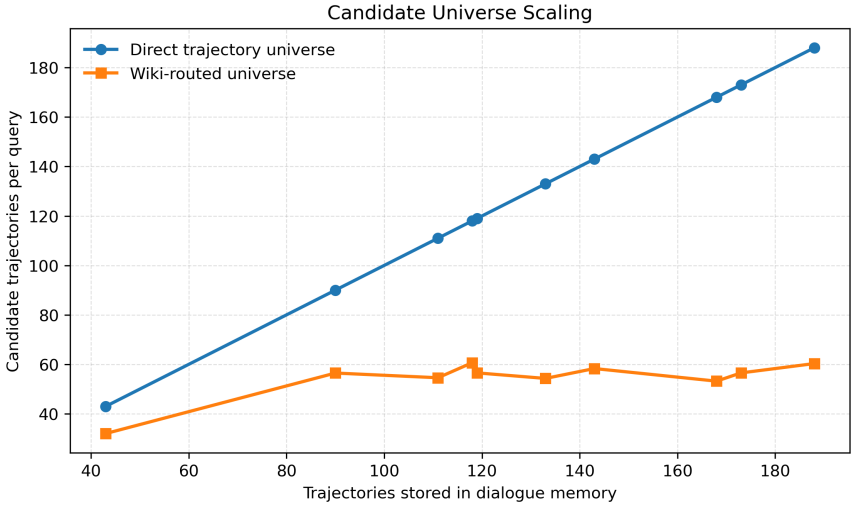

Figure 9: Candidate universe scaling. Wiki routing keeps the trajectory candidate universe substantially smaller than direct retrieval over all stored trajectories.

Table 8: Answer-support diagnostics on LoCoMo. Metrics are offline proxies computed from recorded support references and gold evidence, not human-verified hallucination labels.

Diagnostic	Value
Evaluated queries	282
Source-supported answer proxy	0.390
Unsupported-answer risk proxy	0.610
Mean retrieved gold-reference coverage	0.615
Mean support gold-reference coverage	0.332
Invalid supporting-reference rate	0.216
Unsupported extra-item count	30
Unsupported count-error count	3
Support refs with no gold overlap	78

Table 9: Failure-localization proxy distribution. The table reports automatically assigned diagnostic stages; no human failure-stage labels are used.

Diagnostic Stage	Queries	Rate
Unsupported overgeneration	133	0.472
Answer synthesis error	76	0.270
Trajectory selection miss	28	0.099
Page routing miss	12	0.043
Grounding miss	1	0.004
Correct	32	0.113

B.4 Limitations and Diagnostic Visibility

We provide additional diagnostics for the limitations discussed in Section 4.8. These diagnostics are computed offline from the LoCoMo run with GPT-4o-mini. They do not introduce extra LLM calls and should be interpreted as observable proxies rather than human-verified audit accuracy. In particular, we do not report true conflict detection accuracy, obsolete-claim handling accuracy, or human auditor time/error rate without external labels.

Source-support and unsupported-answer diagnostics. Table 8 summarizes answer-support diagnostics. TrajWiki records supporting references for final answers and compares them with gold source references. The source-supported answer proxy is 0.390, while the unsupported-answer risk proxy is 0.610. The mean retrieved gold-reference coverage is 0.615, but the mean support gold-reference coverage is 0.332, indicating that gold evidence is often retrieved but not always preserved as final answer support. This supports the diagnostic value of TrajWiki: it can distinguish retrieval availability from answer-stage grounding failures.

Failure localization. Table 9 reports the distribution of automatically localized failure stages. These stages are derived from recorded provenance, retrieval metadata, gold evidence coverage, and answer-support diagnostics. They are useful for error analysis, but should not be read as failure-localization accuracy because no human stage labels are provided. The largest category is unsupported overgeneration, followed by answer synthesis error. This indicates that many failures occur after relevant evidence is available, rather than only at the retrieval stage.

Independent evaluation and answer-level ablations. The original judge-based results have a possible evaluator self-preference confound: GPT-4o-mini and Qwen3-8B judged outputs from their own model settings, while Qwen3-32B was evaluated by the same-family Qwen3-8B model. This concern directly affects judge-based accuracy. F1 and BLEU-1 use the separate canonicalized scoring procedure described in Appendix B.1.

For the follow-up evaluation, a method-blinded Claude Sonnet 4.6 judge assessed outputs from the full system, ablated variants, and reference systems. This judge was not used for memory construction or answer generation, and variant identities were omitted. This avoids using the generating model or its model family as the judge in the follow-up, but does not remove other automated-evaluation biases or retrospectively change the evaluation of Table 1.

Table 10: **Answer-level ablations on 200 LoCoMo multi-hop queries.** The full and ablated settings use frozen memory stores and a common 32K input-plus-output cap. Score differences are relative to Full TrajWiki, with 95% confidence intervals from 10,000 paired query-bootstrap draws. Strict accuracy and gold-reference coverage are percentages; input tokens are mean generation inputs, not full-pipeline deployment costs. Source-reference coverage is not reported for the summaries-only condition (N/A).

Setting	Judge Score	Δ vs. Full [95% CI]	Strict Acc.	Gold-ref. cov.	Input tokens
Full TrajWiki	42.0	—	20.5	60.1	19.3K
No Wiki Routing	31.3	-10.8 $[-16.0, -5.5]$	15.5	42.4	30.9K
Latest Snapshot Only	37.5	-4.5 $[-8.5, -0.5]$	16.0	61.5	10.0K
Wiki Summaries Only	24.8	-17.3 $[-22.8, -11.8]$	10.0	N/A	4.5K

The original counterfactual ablations measured retrieval and context proxies without rerunning answer generation. We therefore reran generation on the same 200 LoCoMo multi-hop queries across the follow-up settings, using GPT-4o-mini, identical answering instructions, and a common 32K-token input-plus-output cap. We report

$$S = 100\,\frac{N_c + 0.5N_p}{N}, \tag{13}$$

where N_c and N_p count answers judged `CORRECT` and `PARTIAL`, respectively. Strict accuracy is $100N_c/N$. Table 10 reports these answer-level results.

The bootstrap intervals quantify query-sampling uncertainty, not variation from repeated stochastic generation. No Wiki Routing bypasses query-time routing while reusing the frozen trajectory store; it is not a reconstruction of the system without Wiki-based organization. Under these conditions, the results support answer-level benefits from Wiki routing and source expansion. Retaining historical snapshots provides a smaller benefit despite comparable aggregate reference coverage, and is not established as necessary for every query.

Simpler systems remain competitive in this follow-up. Matched Full Context achieved a JudgeScore of 39.0, and the confidence interval for its paired difference from Full TrajWiki included zero. Naive Dense RAG scored 40.3 with only 1.5K mean input tokens. Thus, these results do not establish universal accuracy or efficiency superiority.

Computational cost and amortization. The observed deployment run used 43.28M tokens, excluding benchmark-only evaluation: 23.71M for memory construction, 10.84M for query-time retrieval, 6.92M for answer generation, and 1.81M for mixed repair and validation. The reported 25.66 hours is accumulated provider-side construction latency, not end-to-end wall-clock time. Reassigning mixed repairs to their corresponding phases yields 24.85M upfront reusable tokens and 18.43M query-side tokens over 282 evaluated queries.

Holding this entire memory collection fixed and using the observed mean query-side cost, amortization over Q queries is

$$\overline{C}_{\mathrm{run}}(Q) \approx \frac{24.85 \times 10^6}{Q} + \frac{18.43 \times 10^6}{282} \quad \text{tokens/query}. \tag{14}$$

At $Q = 282$, the mean is approximately 153.5K tokens per query. The approximately 2.49M upfront tokens per dialogue sample must instead be amortized over the queries served by that individual dialogue, not over the run-wide query count.

This accounting demonstrates reuse under a fixed-memory assumption, not a break-even point against other systems. We did not collect comparable end-to-end cost ledgers for all baselines, including Mem0 and A-MEM, and therefore cannot determine whether the gains justify the additional token or monetary cost. The measured efficiency benefit is a $2.35\times$ reduction in query-time trajectory candidates, from 130.4 to 55.6 on average. It is not evidence of lower total cost.

Conflict, obsolete-claim, and audit-packet diagnostics. TrajWiki also records claim lifecycle state and compact audit packets. In this run, conflict-related context was observable for 0.535 of queries, deprecated claims were suppressed in rendered context with rate 1.000, and the deprecated-leakage

Table 11: Auditability diagnostics. Conflict and obsolete-claim metrics are observable proxies. Audit-packet size measures the compactness of evidence made available for inspection.

Diagnostic	Value
Conflict exposed rate	0.535
Deprecated-claim suppression rate	1.000
Deprecated leakage proxy	0.128
Obsolete-answer risk proxy	0.128
TrajWiki mean audit source count	195.1
Full-context proxy source count	594.7
TrajWiki mean audit packet tokens	11.34K
Full-context proxy audit tokens	20.33K
TrajWiki mean audit claim count	315.2

Table 12: Fallback and repair diagnostics on LoCoMo with GPT-4o-mini. Extra tokens and calls quantify the overhead introduced by fallback, repair, and deterministic validation paths.

Stage	Events	Fallbacks	Repairs	Risk Events	Extra Tokens
Answer synthesis / validation	554	0	69	330	25.10M
Repair phase	564	0	1408	158	1.84M
Semantic metrics	848	1	0	1	0.00M
Unknown fallback events	156	7	0	0	0.00M
Total	2122	8	1477	489	26.93M

proxy was 0.128. These numbers show that TrajWiki exposes conflict and obsolete-claim signals, but they are not substitutes for labeled conflict or obsolete-handling accuracy.

The audit packet provides a compact unit for human or post-hoc inspection. Compared with a full-context audit proxy, TrajWiki reduces the mean audit source count from 594.7 to 195.1 and the mean audit packet size from 20.33K to 11.34K estimated tokens. This suggests that the provenance structure can make post-hoc inspection more focused, although actual human audit time and error rate remain future work.

Fallback and repair overhead. A separate limitation is structured-output dependence. Table 12 summarizes fallback and repair diagnostics. Even with GPT-4o-mini, TrajWiki records substantial repair activity, especially around answer synthesis, validation, and claim preservation. These paths are useful for robustness and diagnosis, but they increase cost and latency. For weaker or non-instruction-tuned models, this dependency is likely to become more severe.

The ledger reports approximately 2,562 extra provider calls, 26.93M extra tokens, and 23.90M ms of accumulated provider-side latency from fallback and repair-related paths. These overheads should not be interpreted as end-to-end failures, but they show that reliable structured generation is an operational requirement for TrajWiki.

Case study: summary-mediated salience drift. A concrete example is query `conv-26_qa_24`:

> **Question:** What does Melanie do to destress?
> **Gold answer:** Running, pottery
> **Model answer:** Melanie engages in various self-care activities to destress, including running, reading, playing the violin, and prioritizing self-care.

The raw evidence supports both required items: `D5:4` states that Melanie signed up for a pottery class and described it as "therapy for me," while `D7:22` states that she had been running farther to de-stress. This was not a pure retrieval miss: the relevant evidence was present in the memory store. The failure occurred because the pottery fact became less salient through snapshot, trajectory, and wiki-level abstraction. The fact survived in source-linked metadata, but it was not foregrounded strongly enough during final answer synthesis.

This case illustrates the main limitation of provenance-aware summarization. Source links make the failure traceable, but they do not guarantee that every source-specific fact remains salient after repeated abstraction. Future work should combine stricter constrained decoding, periodic source-summary consistency audits, and final-answer verification against raw source evidence rather than generated summaries alone.

C Prompts

This appendix lists the main prompts used by TrajWiki. For tasks using structured output, the prompt is paired with a schema through the provider's structured-output interface; we show the natural-language instruction portion here.

C.1 Memory Construction Prompts

Episodic Claim Extraction.

```
TASK=EPISODIC_CLAIM_TEXT_EXTRACT_STRUCTURED
Rewrite the current exchange into readable, source-grounded atomic claim sentences.

Return structured data only through the provider's structured output channel.

Rules:
- Use only facts explicitly stated in the raw messages.
- Resolve speaker names from the raw source lines. If the raw line says "(Melanie):
    my son", write "Melanie's son", not "Caroline's son", "the user", or "the
   assistant".
- claim text must be a complete readable factual sentence, not a transcript
   fragment or broad summary.
- For list-like utterances, include item-level claims for every stated item.
- Preserve exact raw surface phrases for names, titles, places, counts,
   instruments, events, and items.
- Do not collapse books, instruments, cities, places, activities, events, painted
   objects, or counts into umbrella-only claims.
- Do not replace concrete raw noun phrases with broader categories: if raw says "
   sunset", keep "sunset", not only "nature-inspired artwork".
- You may include broader context, but the original source phrase must remain
   visible in at least one claim.
- supporting_quote must be copied from the raw source message and must support the
    claim.
```

Claim Signal Extraction.

```
TASK=CLAIM_SIGNAL_EXTRACT_STRUCTURED
Extract validated retrieval and display signals from readable claims and source
   snippets.
Return structured data only through the provider's structured output channel.

Rules:
- Every surface/value/value_span/display value must be copied from a supplied
   claim or source snippet.
- Do not invent normalized names or semantic replacements.
- Do not output transcript fragments, greetings, acknowledgements, or sentence
   opener fragments.
- Prefer concise useful retrieval/display signals over exhaustive noisy spans.
```

Trajectory Matching.

```
TASK=TRAJECTORY_MATCH_STRUCTURED
Decide whether the new memory continues one of the shortlisted candidate
   trajectories.
Return structured data only through the provider's structured output channel.

Rules:
- Choose CONTINUE only if the new memory clearly belongs to the same evolving
   trajectory as one candidate.
- Each candidate includes a trajectory summary for the whole thread and a latest
   update note for only the newest step.
- Use the trajectory summary as the primary signal; use the latest update note
   only as recent supporting context.
- Do not require the new memory to resemble only the latest update if the broader
   trajectory summary clearly matches.
- Same person, broad topic, or generic words such as community/support/art are not
    enough to CONTINUE.
- CONTINUE requires the same evolving event, project, item, place, count, or fact
   thread.
- Choose NEW when the memory is about a different item, place, event, count, or
   topic cluster under the same person unless it explicitly updates the existing
   thread.
- selected_candidate must be null when decision is NEW.
- selected_candidate must be one of the provided candidate labels when decision is
    CONTINUE.
- rationale should briefly explain the shared evolving trajectory or why no
   candidate fits.
- Do not say candidates are merely related or similar.
```

C.2 Memory Wiki and Retrieval Prompts

Trajectory Retrieval Summary.

```
TASK=TRAJECTORY_RETRIEVAL_SUMMARY
Create a markdown retrieval summary for ONE episodic trajectory using only the
   supplied grounded internal memory state.

Goal:
- produce a retrieval-oriented synopsis for coarse trajectory selection
- preserve exact names, items, and specific facts
- keep older but query-relevant facts visible even when later snapshots shift
   topic
- surface uncertainty or conflicts explicitly instead of hiding them

Required headings:
## Profile / Stable Facts
## Item Sets / Named Entities
## Relations / Temporal Updates
## Conflicts / Uncertainty

Rules:
1. Use only the supplied claims, facets, exact terms, and recent snapshot notes.
2. Preserve exact titles, books, recipes, instruments, symbols, places, and named
   items.
3. Keep list items explicit; do not collapse them into broad themes.
4. Do not let the newest update erase older item sets, places, events, dates, or
   counts from the trajectory.
5. Do not replace concrete source phrases with broad categories: keep "sunset"
   visible even if you also mention "artwork".
6. Mention contradictory or uncertain facts under Conflicts / Uncertainty instead
   of resolving them yourself.
7. Do not add commentary, preamble, or conclusions outside the markdown sections.
```

```
8. Exclude filler words, acknowledgements, discourse markers, and casual fragments
    from the useful retrieval content.
```

Wiki Page Planning.

```
TASK=WIKI_PAGE_PLAN
Plan a compact sample-level wiki graph from the provided candidate seed manifest.

Return markdown only with the exact heading:
## Pages

Then emit one bullet per page:
- page_type=<index/entity/topic/inventory> | title=<title> | slug=<slug> |
  trajectories=<comma-separated trajectory ids> | entities=<comma-separated
  entities> | links=<comma-separated page slugs>

Rules:
1. Always include exactly one index page.
2. Treat each input block as a candidate seed that already groups related
  trajectories; refine or select from those seeds instead of inventing broad new
  groupings.
3. Create entity pages when a person/entity appears across multiple trajectories.
4. Create inventory pages for list/count/item-heavy facts such as activities,
  books, recipes, instruments, symbols, places, wins, dogs, or repeated events.
5. Create topic pages only when they add navigation value beyond entity or
  inventory pages.
6. Keep pages compact and specific; avoid broad pages that overlap heavily with
  existing entity or inventory coverage.
7. Prefer explicit evidence coverage over narrative grouping.
8. Use trajectory historical evidence cards as first-class evidence; do not rely
  only on a trajectory's latest summary.
9. If a card exposes a specific item, place, event, count, or source anchor, cover
    it in a non-index page whenever possible.
```

Wiki Page Compilation.

```
TASK=WIKI_PAGE_COMPILE
Compile one wiki page from the supplied page metadata, representative trajectory
  summaries, and historical trajectory evidence cards.

Return markdown only with these headings:
## Overview
## Key Facts
## Items / Counts
## Linked Trajectories
## Conflicts / Uncertainty

Rules:
1. Use only supplied linked trajectory ids, dominant entities, exact terms, facet
  values, representative trajectory summaries, and historical evidence cards.
2. Keep exact names, items, places, counts, and titles.
3. For inventory pages, favor explicit item lists over prose.
4. Preserve enumerated items and counts explicitly; do not collapse them into
  broad narrative summaries.
5. Do not replace a concrete source phrase with a broader category: if an evidence
    card contains "sunset", keep "sunset", not only "nature-inspired artwork".
6. Do not add speculative links or inferred facts not stated in the supplied
  inputs.
7. In Items / Counts, use only readable display items, readable counts, and
  readable key facts.
```

```
8. If a latest summary and historical card emphasize different facts, present the
   historical facts in their own concrete bullets instead of dropping them.
9. Do not dump Dominant exact terms verbatim; ignore any low-confidence or
   fragment-like signal.
10. Do not write placeholder descriptions such as "Not provided", "Unknown", "N/A",
    "None provided", "No specific key facts", or "No explicit items".
11. Describe a linked trajectory only when its representative summary or evidence
   card is supplied; otherwise leave the trajectory id without a fabricated
   description.
12. The system may rewrite the final Linked Trajectories section deterministically,
    so keep that section concise.
```

C.3 Answer Generation and Evaluation Prompts

LoCoMo Evidence Synthesis.

```
TASK=LoCoMo_ANSWER_EVIDENCE_SYNTHESIS
Legacy structured LoCoMo evidence synthesis.

Input contains a LoCoMo question followed by retrieved memory/wiki/source evidence.


Rules:
1. Use only retrieved evidence. If it cannot answer the question, set can_answer=
   false, leave final_answer empty, and explain the gap in abstain_reason.
2. Preserve exact source terms for names, titles, places, dates, counts,
   activities, and items.
3. supporting_source_refs must use only visible refs such as D8:4, and each ref
   must support the requested answer family.
4. Count distinct completed events only. Exclude future plans, intentions,
   reactions, duplicate mentions, general hobbies, and uncertain candidates.
5. Count final_answer must be natural language, not a bare number; use an exact
   count only when complete evidence supports it, otherwise state a retrieved-
   evidence lower bound.
6. Date/time answers must use source line date=... fields, snapshot Timestamp
   lines, and the ## Temporal Anchors block; resolve terms such as "yesterday" to
   the grounded date when available.
7. For lists/events/preferences, include every in-scope source-backed specific
   item and avoid scope-mismatched extras.
8. For bridge/alias facts, answer with the concrete source-backed value, e.g. "
   West County" rather than "old area".

Return structured data only. If structured output is unavailable, return only a
   JSON object matching the same fields.
```

LoCoMo Judge.

```
Evaluate the candidate answer against the question and gold answer.

Equivalence Policy:
- Allow semantic equivalence, near-synonyms, ordering/format variants, harmless
   modifiers, and concise or longer phrasing with the same meaning.
- Accept date variants for the same time point, e.g. "May 7th" and "7 May"; an
   unanchored relative date such as "last Tuesday" is PARTIAL, and a clearly wrong
   date is INCORRECT.
- Treat source-backed aliases as covered: "LGBTQ support group" covers "support
   group"; "LGBTQ pride parade" covers "pride parade"; "shared her journey at a
   school event and encouraged students" covers "school speech"; "mentorship
   program for LGBTQ youth" covers "mentoring program".
```

```
- Harmless same-category extras are allowed unless they change the requested scope,
    contradict the gold answer, or replace a required item.

Strict Cases:
- Counts are strict: "Twice", "2 times", and "two confirmed rejections" are
   equivalent, but an unqualified wrong count is INCORRECT.
- A retrieved-evidence lower bound such as "The retrieved evidence confirms one
   rejection" is PARTIAL when the gold requires more.
- Missing a required list item is PARTIAL; for example, omits "school speech" from
    "pride parade, school speech, support group".
- Wrong entity, polarity, count, time, or place is INCORRECT.

Verdicts:
- CORRECT: all required gold facts/items are covered and no contradiction changes
   the answer.
- PARTIAL: at least one required fact is covered, but required content is missing
   or qualified as a lower bound.
- INCORRECT: no required fact matches, or the answer reverses/changes the core
   meaning.
```

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and the introduction summarizes our main contributions.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: This paper cover a section of the limiations.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: N/A

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Both code and datasets are available.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide the code link in the abstract

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We cover all the details in the paper.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper reports point estimates for the main experimental results, but does not currently include error bars, confidence intervals, or statistical significance tests. Repeated full LLM evaluations are costly due to API usage, so we did not run multiple independent trials or report uncertainty estimates in the current version.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: It could be found in the experimental part.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [N/A]

   Justification: N/A

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [No]

    Justification: The current version discusses several technical limitations, including cost, reliance on structured LLM outputs, and summary-mediated salience drift. However, it does not yet adequately discuss potential negative societal impacts. Since the proposed system stores and updates long-term conversational memories, including potentially sensitive personal or medical information, it may raise risks related to privacy, user consent, data retention, deletion, access control, profiling, and propagation of outdated or incorrect memories. But our paper only presents a novel memory system for LLM agents, although different LLM agents may have different social impacts, which is beyond the scope of our work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Their contribution has already been properly acknowledged and credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL when possible.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [No]

    Justification: N/A

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: Our paper proposes an Agent Memory framework that utilizes multiple pre-trained LLMs and OpenAI's LLM Apis, which we have explained in detail in the paper.

    Guidelines:

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