STREAMING MEMORY BENCHMARK: STAGE-LEVEL DIAGNOSIS WITH EVIDENCE DEPENDENCY CONTROL

Guanming Liu[1,4], **Haoran Yin**[1], **Tianchen Li**[1], **Sikuan Yan**[2], **Hongru Wang**[3]
Baian Chen[1], **Xiaoteng Ma**[1]
[1]Mindlab Ltd., https://macaron.im/mindlab
[2]Ludwig-Maximilians-Universität München
[3]University of Edinburgh
[4]Fudan University

ABSTRACT

Memory in deployed LLM agents operates under a streaming regime where evidence arrives incrementally, context is bounded, and every pipeline stage incurs recurring latency and token cost. Yet existing benchmarks evaluate memory statically, providing full history at once and reporting only end-to-end accuracy, making it difficult to (i) attribute failures and costs to specific pipeline stages, and (ii) verify that correct answers truly depend on the memory system rather than in-context extraction or inference shortcuts. We propose a streaming evaluation framework that structures user–agent conversations into streaming episodes with explicit **Evidence–Query** dependencies and evaluates memory through a four-stage pipeline (**Formation**, **Management**, **Retrieval**, **Application**) with stage-level accuracy and efficiency metrics. We further show that simply converting existing static benchmarks into a streaming format is insufficient: retrieval and application accuracy can diverge substantially, indicating that some tasks remain solvable without faithfully retrieving the intended evidence. To address this leakage, we construct StreamMemBench, a natively streaming benchmark with per-episode evidence boundaries and evidence-linked distractors that ensure correct answers require the memory pipeline. Across five memory systems and three datasets, we find that formation accuracy saturates while efficiency differs by an order of magnitude, and that retrieval–application divergence serves as a reliable diagnostic signal for evaluation leakage.

1 INTRODUCTION

When an LLM agent serves a user over weeks of interaction, its memory system must continuously encode new facts, maintain a growing store, retrieve relevant items on demand, and use them to produce answers. Information arrives incrementally, the active context is bounded, and every stage of this pipeline adds recurring latency and token cost (Wang et al., 2025). Memory, in deployment, is inherently streaming.

Yet the benchmarks used to evaluate memory systems Hu et al. (2025b); Maharana et al. (2024); Jiang et al. (2025) do not reflect this reality. Most provide the full conversation history at once and report only end-to-end accuracy, introducing two forms of evaluation failure. First, a single aggregate score offers no way to attribute failures or costs to specific pipeline stages. Concurrent efforts toward streaming evaluation (Wei et al., 2025; Hu et al., 2025a) recognize this problem, yet still offer limited stage-level diagnosis and cost attribution. Second, when full history remains in context, correct answers may come from direct extraction or inference shortcuts rather than from the memory pipeline. A benchmark that does not control the dependency between evidence and queries can show improvements that do not transfer to deployment.

We propose a streaming evaluation framework that makes memory evaluation stage-diagnosable (Figure 1). Grounded in cognitive psychology's multi-phase memory model Kliegel et al. (2002); Nyberg & Eriksson (2016); Khan et al. (2025), also adopted by (Hu et al., 2025b; Chen et al., 2025), we structure interactions into **streaming episodes** with explicit **Evidence** (E) and **Query** (Q) de-

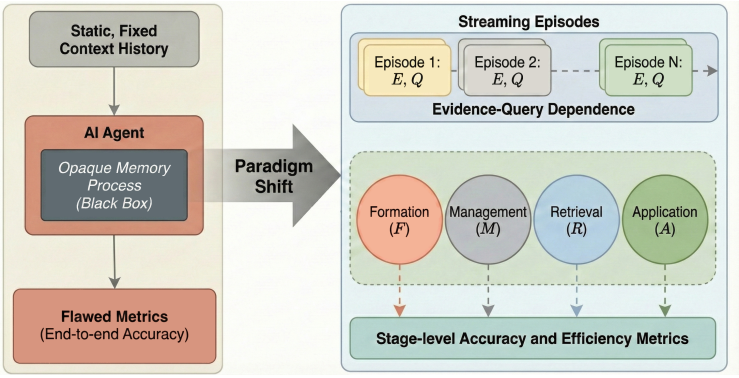

Figure 1: The paradigm shift from static, opaque evaluation to streaming evaluation with a transparent four-stage pipeline.

pendencies, and decompose evaluation into four stages: **Formation**, **Management**, **Retrieval**, and **Application**, each with accuracy and efficiency metrics. We apply this protocol to existing static benchmarks via a streaming transformation, but find that retrieval and application accuracy can diverge substantially, indicating that some tasks remain solvable without faithful evidence retrieval. Since statically authored datasets were not designed with evidence dependencies in mind, their structure still permits shortcut solutions. This motivates **StreamMemBench**, a natively streaming benchmark with explicit evidence boundaries and evidence-linked distractors that ensure correct answers require the memory pipeline (Figure 2). Our contributions:

- A four-stage streaming evaluation protocol (Formation, Management, Retrieval, Application) with per-stage accuracy and efficiency metrics, enabling stage-level attribution of failures and costs.
- A streaming transformation for existing benchmarks and **StreamMemBench**, a natively constructed benchmark with evidence dependency control that prevents shortcut solving.
- Empirical findings across five memory systems and three datasets: formation accuracy saturates while efficiency differs by $\sim$10$\times$; retrieval–application divergence reveals cases where correct answers do not require the complete memory pipeline; and delayed-query analysis shows application degrades less than retrieval, reinforcing the need for stage-level evaluation.

2 Streaming Evaluation with Evidence Dependency

2.1 Core Principle

At the core of memory lies a simple premise: every valid recall should be grounded in a specific antecedent, i.e., **Evidence**. In real deployment, evidence arrives incrementally rather than as a preloaded batch. We mirror this by restructuring evaluation into **Streaming Episodes**, where each episode provides Evidence (E) that supports one or more future Queries (Q). This design addresses the two gaps identified above: by decomposing evaluation into stages, we gain *stage-level attribution*; by separating evidence introduction from query time, we gain *evidence dependency control* over what must pass through the memory pipeline.

This transformation enforces a strict **Streaming Dependency**, formalized as $S = \langle (E_1, \mathcal{Q}_1), \ldots, (E_t, \mathcal{Q}_t), \ldots \rangle$, where each episode t provides new evidence E_t and is associated with a set of queries $\mathcal{Q}_t$. Making evidence boundaries explicit allows us to control when information is introduced and when it is queried, while still approximating real-world interaction patterns.

2.2 Streaming Transformation of Existing Benchmarks

We apply streaming transformation to two datasets. **LoCoMo** Maharana et al. (2024) consists of two-person dialogues; after transformation, evidence is densely packed ($\sim$5 turns/episode), testing

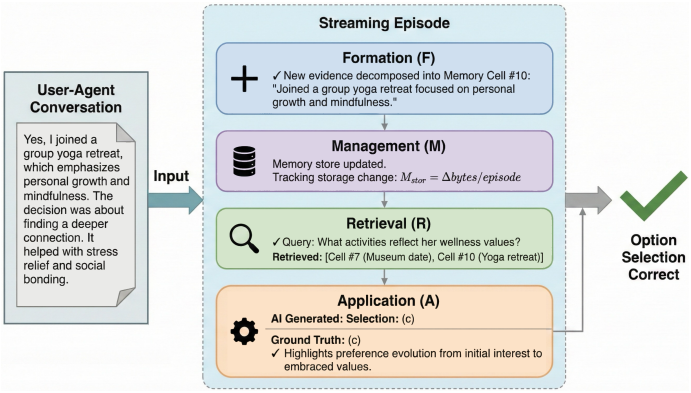

Figure 2: Operational workflow of the four-stage evaluation protocol (Formation, Management, Retrieval, Application) on a concrete example.

Table 1: Dataset characteristics. LoCoMo and PersonaMem are transformed from static benchmarks; StreamMemBench is natively constructed.

Dataset	Focus	Turns/Ep.	Evid./Ep.	Core Challenge
LoCoMo	Fact Reasoning	$\sim$5	2.06	Frequent updates, high-density factual evidence
PersonaMem	Preferences	$\sim$20	2.84	Long-range tracking, sparse preference evolution
StreamMemBench	Preferences	$\sim$25	5.1	Preference shifts, cross-episode reasoning, persona-centered

temporal ordering and short-range multi-hop reasoning. **PersonaMem** Jiang et al. (2025) consists of human-AI dialogues; after transformation, evidence is sparser ($\sim$20 turns/episode), with categories involving preference evolution and memory utilization, placing greater demands on retrieval and application. Table 1 summarizes these two datasets alongside StreamMemBench, introduced below.

Beyond transforming existing benchmarks, we construct **StreamMemBench**, a natively streaming dataset with strict per-episode evidence boundaries, ensuring valid memory evaluation (Figure 3). Construction starts from a structured persona whose preference domains form a finite *preference pool*, partitioned into three states: *unrevealed*, *revealed*, and *flipped*. For each episode, an event planner selects focus facts from the pool and generates a scenario; a conversation generator produces dialogue so that each episode contains only the designated evidence pieces; and a QA(Question-Answer) generator produces evidence-linked MCQ (multiple-choice question) distractors constructed from partial or mis-composed evidence (e.g., mixing fields across entities or using pre-flip preferences). Generator–judge validation loops enforce evidence coverage, scope control, and distractor quality. Cross-episode QA with evidence spanning $\geq$2 sessions probes integration capabilities such as entity binding and conflict resolution. Full details are in Appendix D.

2.3 Protocol Evaluation for Memory

To enable stage-level attribution, we design four evaluation protocols corresponding to the complete memory lifecycle (Table 2). Each protocol targets a specific pipeline stage, reporting accuracy and/or efficiency so that failures and costs can be attributed to their source. For Formation, Retrieval, and Application, we evaluate both accuracy and efficiency; for Management, we evaluate efficiency only, as management actions (e.g., compressing and conflict handling) are strategy-dependent and their effects are better captured by downstream Retrieval/Application accuracy. Figure 2 illustrates the operational workflow with a concrete example.

For protocols with correctness evaluation ($s \in \{F, R, A\}$), we measure accuracy (s_{acc}, binary 0/1 automatic judgment) and token cost (s_{tok}). In our implementation, Formation uses a model-based judgment of whether the evidence is semantically captured in the newly formed memory output. Retrieval uses a model-based support judgment over the gold evidence and retrieved memories, measuring whether the retrieved set covers the gold evidence needed to answer the query. Applica-

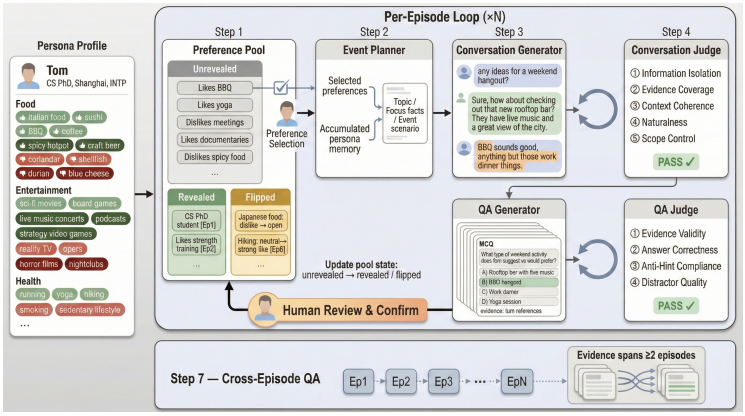

Figure 3: The StreamMemBench construction pipeline with preference pool management, generator-judge validation, human review, and cross-episode QA generation.

Table 2: Protocol specifications. F=Formation, M=Management, R=Retrieval, A=Application. Management evaluates only efficiency.

Protocol	Target	Accuracy	Efficiency
F	evidence encoded in memory	automatic existence check	latency, formation tokens
M	storage dynamics	–	relative storage change (bytes/episode)
R	gold evidence (one or more turns) retrieved	automatic semantic support check	latency, retrieved context tokens
A	question answered correctly	automatic answer check	latency, answer tokens (including reasoning content)

tion uses exact match for multiple-choice items and model-based answer judgment for open-ended items. Management evaluates only efficiency through M_{stor}, the relative change in stored memory size (in bytes) per episode, tracking how the system manages additions, updates, and consolidations over time.

3 Experiments

We evaluate four memory systems plus an LLM baseline. Each system differs in how it encodes information, retrieves context, and generates responses: EverMemOS Hu et al. (2026) uses event-centric atomic facts with hybrid retrieval; Mem0 Chhikara et al. (2025) uses attribute-centric facts with vector retrieval; MemOS Li et al. (2025) uses structured KV metadata; and MemoryOS Kang et al. (2025) stores raw dialogue with hierarchical memory. Full system specifications are in Table 5 (Appendix). We use Gemini-3-flash-preview as the backbone model for all memory systems and for model-based correctness checks when needed. We evaluate 50 episodes per dataset on LoCoMo and PersonaMem, and 20 episodes on StreamMemBench. Episodes are streaming interaction units rather than individual test questions: on the exact subsets used in the main table, this corresponds to 444 evaluable $F/R/A$ instances on LoCoMo, 231 on PersonaMem, and 306 on StreamMemBench. Appendix C reports a human audit of the automatic labels using approximately 20% samples per metric.

3.1 Overall Results

Table 3 consolidates all metrics across protocols. We organize findings around the two properties introduced above: stage-level attribution (diagnosis and efficiency) and evidence dependency control (whether correct answers depend on the memory pipeline).

Formation saturates in accuracy but differs sharply in efficiency. All systems achieve near-perfect formation accuracy, suggesting that evidence encoding is rarely a correctness bottleneck in

Table 3: Main results across protocols. †MemoryOS stores raw dialogue ($F_{\text{acc}} = 1.0$ by definition).

		Formation		Management	Retrieval		Application		Latency (s)		
System	**Dataset**	F_{acc}	F_{tok}	M_{stor}	R_{acc}	R_{tok}	A_{acc}	A_{tok}	F	R	A
LLM Baseline	LoCoMo	0.974	119.9	433.3	0.920	145.5	0.717	579.6	8.93	0.41	2.95
EverMemOS	LoCoMo	1.000	277.6	1328.2	0.832	141.8	0.779	3839.3	7.70	0.66	7.62
MemOS	LoCoMo	1.000	159.9	745.3	0.956	421.8	0.841	2393.4	6.59	0.87	4.16
Mem0	LoCoMo	0.979	93.6	135.9	0.938	154.8	0.805	1836.2	12.53	0.27	2.58
MemoryOS†	LoCoMo	1.000	240.6	1085.4	0.611	1195.2	0.531	3926.6	36.84	2.53	4.64
LLM Baseline	PersonaMem	0.919	202.7	832.8	0.537	181.5	0.744	889.2	11.52	0.32	2.26
EverMemOS	PersonaMem	1.000	510.1	2572.0	0.465	199.7	0.802	3254.8	11.51	1.02	8.13
MemOS	PersonaMem	0.988	278.4	1419.0	0.570	450.7	0.767	1671.7	8.20	0.91	3.20
Mem0	PersonaMem	1.000	167.6	556.2	0.616	208.6	0.756	1562.8	14.23	0.57	2.16
MemoryOS†	PersonaMem	1.000	2348.5	6403.5	0.419	1598.6	0.558	2908.3	27.99	0.22	1.89
LLM Baseline	StreamMemBench	0.971	363.4	765.4	0.931	164.8	0.843	1385.3	52.29	1.79	8.35
EverMemOS	StreamMemBench	0.990	698.0	806.5	0.922	247.0	0.882	7069.5	20.40	1.47	8.56
MemOS	StreamMemBench	0.990	410.9	427.2	0.922	675.1	0.902	4360.8	12.44	3.55	15.11
Mem0	StreamMemBench	0.990	265.1	212.0	0.902	132.4	0.824	5266.5	39.30	1.26	7.43
MemoryOS†	StreamMemBench	1.000	1203.6	1651.5	0.578	1520.1	0.618	6400.4	98.39	6.81	9.75

our setting. The dominant differences are in efficiency: token cost, storage growth, and latency vary by an order of magnitude across systems. Attribute-centric extraction (Mem0) and key-value structuring (MemOS) yield compact representations with minimal per-episode M_{stor} growth, while hierarchical raw-dialogue approaches (MemoryOS) incur the highest cost. Across all systems, formation also dominates latency relative to retrieval and application, highlighting a practical deployment bottleneck.

Retrieval–application divergence reveals evaluations solvable without faithful evidence retrieval. The gap between retrieval and application accuracy offers a direct signal for evidence dependency control. On PersonaMem, we consistently observe $R_{\text{acc}} < A_{\text{acc}}$ across systems, indicating that some items remain solvable without retrieving the designated evidence. In contrast, StreamMemBench, with strictly controlled evidence boundaries and evidence-linked distractors, produces a stable $R_{\text{acc}} > A_{\text{acc}}$ regime, showing that retrieving the right evidence is necessary but not sufficient for effective memory use.

When retrieval fails, application may still succeed via high-level summaries. MemoryOS maintains a user profile that summarizes persona-level preferences. This profile often occupies a top-k retrieval slot with high similarity scores but contains aggregated information rather than the designated conversational evidence, which lowers retrieval recall. Nevertheless, the same summary can provide sufficient persona context for the backbone model to infer the correct option, improving application accuracy. Together with the R–A gap analysis above, this suggests that retrieval and application should be co-designed so that retrieved content matches what the downstream stage can reliably use, rather than relying on implicit inference.

3.2 Category-Level Divergence Analysis

Table 4 drills down on the aggregate R–A patterns by comparing R_{acc} and A_{acc} in two representative PersonaMem categories, illustrating how different error sources produce opposite gaps. In *suggest_new_ideas*, all systems exhibit $R > A$: retrieval often surfaces the relevant evidence, but the application stage fails to reliably use it when options are semantically close. In *track_pref_evolution*, the opposite pattern emerges ($A > R$): systems answer correctly despite retrieval failure because models leverage nearby preference descriptions or infer from general conversation context. On the LLM baseline,

Table 4: Category-level divergence between retrieval and application (PersonaMem).

Category	**System**	R_{acc}	A_{acc}	**Gap**
suggest_new_ideas	LLM Baseline	0.78	0.44	+0.34
	EverMemOS	0.78	0.22	+0.56
	MemOS	0.91	0.45	+0.46
	Mem0	0.89	0.33	+0.56
	MemoryOS	0.83	0.25	+0.58
track_pref_evolution	LLM Baseline	0.47	0.63	−0.16
	EverMemOS	0.37	0.79	−0.42
	MemOS	0.42	0.68	−0.26
	Mem0	0.58	0.68	−0.10
	MemoryOS	0.32	0.42	−0.10

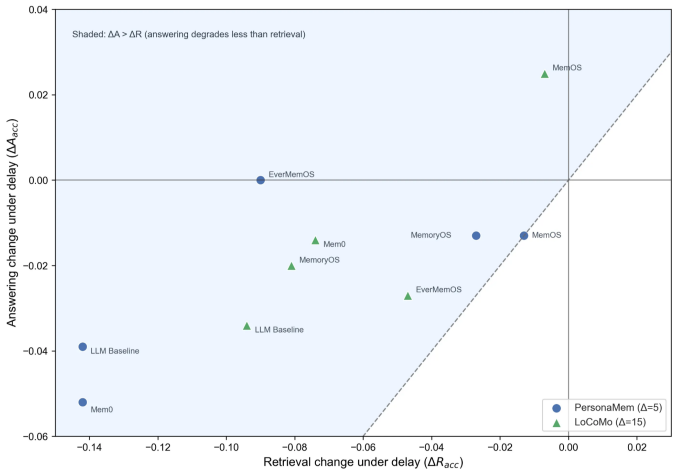

Figure 4: Performance changes under episode delay. The diagonal represents equal degradation in retrieval and application. All systems fall into the shaded region ($\Delta R < 0$, $|\Delta R| > |\Delta A|$), indicating retrieval degrades slightly while application remains stable.

33% of correct answers (28/86) occur without successful retrieval ($\Pr(A = 1 \mid R = 0) = 72\%$), indicating that many questions can be solved through contextual inference rather than faithful memory retrieval.

These findings reveal a fundamental limitation of current datasets: the R–A gap indicates that end-task accuracy masks whether the system truly retrieved the designated evidence or inferred it from context. This motivated StreamMemBench's design with strict evidence boundaries and evidence-linked distractors that cannot be resolved without the exact designated evidence.

3.3 Episode-Delay Analysis

We test temporal robustness by delaying queries Δ episodes after evidence appears. We choose $\Delta = 15$ for LoCoMo and $\Delta = 5$ for PersonaMem to approximate comparable temporal distances in terms of turns. Figure 4 shows that all systems maintain stable performance: retrieval accuracy drops by $<2\%$ and application accuracy remains stable.

Two patterns emerge. First, MemOS on LoCoMo achieves $+1.1\%$ A_{acc} under delay. We attribute this to its textual memory design: when later episodes contain semantically similar expressions, consolidation integrates them with earlier memories, yielding more complete representations. Second, application consistently degrades less than retrieval. This asymmetry suggests that some questions can be answered through inference even when retrieval fails, reinforcing the need for evidence dependency control.

4 Conclusion

We presented a streaming evaluation framework that makes memory evaluation stage-diagnosable and verifiable. Our four-stage protocol (Formation, Management, Retrieval, Application) enables stage-level attribution of both errors and costs, while StreamMemBench enforces evidence dependency control to ensure that correct answers require the memory pipeline rather than context-level shortcuts. Across five systems and three datasets, formation accuracy saturates but efficiency varies by an order of magnitude, making cost the primary differentiator for deployable memory. Retrieval–application divergence reveals that some evaluations can be solved without faithful evidence retrieval, and this asymmetry persists under delayed queries, confirming the need for evidence dependency control and stage-level evaluation. Our results argue that memory should be evaluated in the regime it is deployed: streaming, cost-sensitive, and stage-diagnosable, so that benchmark progress reflects real-world agent performance.

REFERENCES

Ding Chen, Simin Niu, Kehang Li, Peng Liu, Xiangping Zheng, Bo Tang, Xinchi Li, Feiyu Xiong, and Zhiyu Li. HaluMem: Evaluating hallucinations in memory systems of agents. *arXiv preprint arXiv:2511.03506*, 2025.

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

Chuanrui Hu et al. EverMemOS: A self-organizing memory operating system for structured long-horizon reasoning. *arXiv preprint arXiv:2601.02163*, 2026.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*, 2025a.

Yuyang Hu et al. Memory in the age of AI agents. *arXiv preprint arXiv:2512.13564*, 2025b.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J. Taylor, and Dan Roth. Know me, respond to me: Benchmarking LLMs for dynamic user profiling and personalized responses at scale. In *Proceedings of the Conference on Language Modeling (COLM)*, 2025.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Adeel Khan et al. From correlation to causation: Understanding episodic memory networks. *Neuroscience Bulletin*, 2025.

Matthias Kliegel, Mike Martin, Mark A. McDaniel, and Gilles O. Einstein. Complex prospective memory and executive control of working memory: A process model. *Psychologische Beiträge*, 44(2–3):303–318, 2002.

Zhiyu Li et al. MemOS: A memory OS for AI system. *arXiv preprint arXiv:2507.03724*, 2025.

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

Lars Nyberg and Johan Eriksson. Working memory: Maintenance, updating, and the realization of intentions. *Cold Spring Harbor Perspectives in Biology*, 8(2):a021816, 2016.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Acting less is reasoning more! teaching model to act efficiently. *arXiv preprint arXiv:2504.14870*, 2025.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, et al. Evo-Memory: Benchmarking LLM agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

A LIMITATIONS

Our evaluation covers five memory systems with small episodes per dataset, prioritizing depth of analysis over breadth. Adapting each system to the streaming protocol requires substantial engineering effort, as internal designs vary significantly. The episode count limits statistical power at fine-grained category levels, though consistent cross-system patterns suggest the findings are robust.

B SYSTEM SPECIFICATIONS

We evaluate four memory systems plus an LLM baseline, covering diverse design choices in memory representation and retrieval. Full specifications are shown in Table 5.

Table 5: System components by pipeline stage.

System	Memory Form	Retrieval	Answer Context
EverMemOS	Event-centric Atomic Fact	Hybrid (BM25 + Vector)	Memcells (CoT)
Mem0	Attribute-centric Fact	Vector (Qdrant)	Fact-based Context
MemOS	Structured KV Metadata	Vector (Qdrant)	Textual Memories
MemoryOS	Raw Dialogue + Hierarchical	Multi-layer Search	Multi-source Context
LLM Baseline	Independent Fact	Vector (Cosine)	Fact-based Context

C HUMAN AUDIT

We conducted a stratified human audit on approximately 20% of the cases for each metric. The corresponding $F/R/A$ sets contain 77/77/77 cases on PersonaMem, 148/148/148 on LoCoMo, and 102/102/102 on StreamMemBench. For each sampled case, we reviewed the evaluation outputs: formed memories for F_{acc}, retrieval support records for R_{acc}, and generated answers for A_{acc}. Table 6 reports automatic positives, human positives, and agreement. Overall agreement is high across all three metrics, with perfect alignment for Application in the sampled cases.

Table 6: Human audit of automatic $F/R/A$ labels. Each row audits approximately 20% of the corresponding metric-specific set.

Dataset	Metric	Total	Audit n	Auto +	Human +	Agree
PersonaMem	F_{acc}	77	15 (19.5%)	9/15	9/15	14/15
PersonaMem	R_{acc}	77	15 (19.5%)	8/15	7/15	14/15
PersonaMem	A_{acc}	77	15 (19.5%)	8/15	8/15	15/15
LoCoMo	F_{acc}	148	30 (20.3%)	25/30	24/30	29/30
LoCoMo	R_{acc}	148	30 (20.3%)	15/30	15/30	30/30
LoCoMo	A_{acc}	148	30 (20.3%)	15/30	15/30	30/30
StreamMemBench	F_{acc}	102	20 (19.6%)	17/20	16/20	19/20
StreamMemBench	R_{acc}	102	20 (19.6%)	13/20	13/20	20/20
StreamMemBench	A_{acc}	102	20 (19.6%)	10/20	10/20	20/20

Protocol-F checks whether the gold evidence is semantically captured by newly formed memory, and Protocol-R checks whether the retrieved memories cover the gold evidence needed to answer the query.

D STREAMMEMBENCH CONSTRUCTION DETAILS

Preference Pool. Each persona defines preference domains (e.g., food, entertainment, health) with typed items carrying polarity (like/dislike) and strength (normal/strong). These items form the preference pool, partitioned into three states: *unrevealed* (not yet introduced), *revealed* (introduced in a past episode), and *flipped* (preference reversed). The pool serves as the single source of ground-truth evidence, ensuring every fact in the dataset is traceable.

Event Planning. Before generating each episode, an event planner selects focus facts from unrevealed preferences and produces a concrete scenario (topic, event, goal, opening). The planner consults the system's accumulated persona memory to guarantee that focus facts do not overlap with already-known information. If the operator selects preferences for flipping, the planner additionally generates a preference change specification (old $\rightarrow$ new, with reason).

Conversation Generation. A conversation generator produces multi-turn dialogue under strict information isolation: the assistant's persona memory is deliberately withheld from the generation

context, so the generated assistant cannot reference anything the user has not said in the current episode. User utterances are constrained to be concise ($\leq$50 characters) and must include at least one correction or disagreement to ensure naturalness.

Conversation Judge. A separate judge model evaluates each generated episode across five dimensions: (1) information isolation, (2) evidence coverage (all focus facts appear in user turns), (3) context coherence, (4) naturalness, and (5) scope control (no extra memorizable information beyond focus facts). If the judge identifies issues, it provides concrete revision suggestions; the generator then revises only the affected turns. This loop runs up to 3 rounds.

QA Generation. For each episode, questions are generated across seven types: *fact_recall*, *preference*, *plan*, *reasoning*, *negation*, *detail_precision*, and *implicit_info*. Distractors are generated separately with type-specific strategies (e.g., negation distractors include the pre-negation state; detail_precision uses adjacent values). Each question references specific dialogue turns as evidence.

QA Judge. A dedicated judge model validates each generated QA pair across four dimensions: (1) evidence validity (the referenced dialogue turns support the question), (2) answer correctness (the labeled answer is unambiguously correct), (3) anti-hint compliance (question stem and distractors do not leak the answer), and (4) distractor quality (wrong options are plausible but clearly incorrect). If the judge identifies issues, it returns specific revision instructions; the QA generator revises only the affected questions. This loop also runs up to 3 rounds.

Cross-Episode QA. After all episodes are finalized, cross-episode questions are generated with evidence spanning $\geq$2 sessions. Six question types probe integration capabilities: *entity_binding* (linking references to the same entity across episodes), *slot_completeness* (aggregating attributes from multiple sessions), *evolution_timing* (ordering preference changes temporally), *slot_mapping* (transferring preferences across related domains), *scenario_change* (adapting to contextual shifts), and *conflict_resolution* (resolving contradictory preferences over time). These questions test whether memory systems can synthesize information across temporally distant interactions, requiring **evidence chaining** where the answer depends on composing multiple pieces of evidence from different episodes.

Construction Console. Figure 5 shows the construction console interface, which provides real-time visualization of persona preference domains and episode-level pool management (revealed vs. unrevealed preferences).

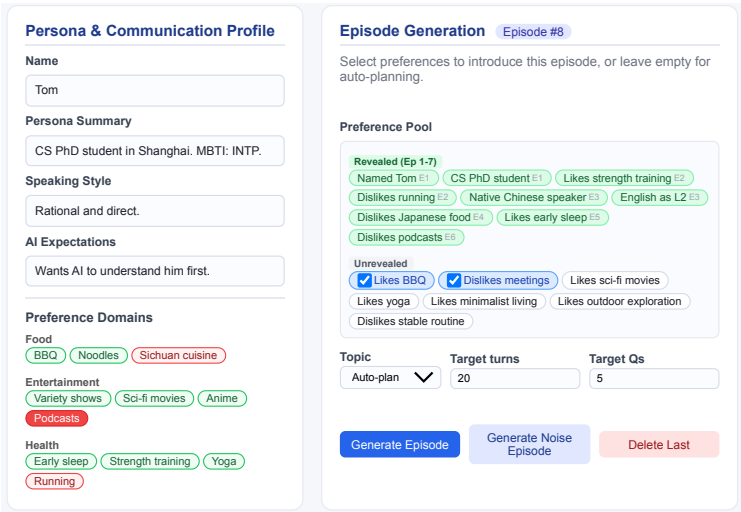

Figure 5: The StreamMemBench construction console. Left: persona with structured preference domains. Right: episode generation with preference pool management, showing revealed (green) and unrevealed preferences.