StreamMemBench: Streaming Evaluation of Agent Memory for Future-Oriented Assistance

Anonymous ACL submission

Abstract

A central role of personal-agent memory is to turn stored information and prior interactions into future-oriented assistance. In daily use, useful cues come from what the agent observes and how the user interacts with the agent, and the agent must carry them forward from the current request to similar future tasks. Existing memory benchmarks usually test dialogue recall or task improvement in isolation, leaving the trajectory from streaming observations to later assistance largely untested. We introduce StreamMemBench, a streaming benchmark that constructs a two-step task sequence around each evidence anchor from EgoLife egocentric streams. The initial task tests evidence use, while the follow-up task tests whether feedback and interaction experience are reused. Four metrics diagnose evidence recall, initial evidence use, feedback incorporation, and follow-up reuse. Experiments with eight memory systems across two backbones show that current systems often fail to use observed evidence or turn feedback into reliable follow-up behavior, even when evidence is stored or feedback is incorporated locally.

1 Introduction

Memory is widely recognized as a foundational capability for personal agents built on large language models (LLMs). Beyond retaining user-specific information such as preferences, habits, relationships, and prior interactions (Hu et al., 2026a; Chhikara et al., 2025; Kang et al., 2025; Li et al., 2025), memory mechanism should also enable this information to be applied to future tasks. However, most existing memory agents are unable to accomplish this goal. This limitation is especially important when personal agents operate over continuous streams. To enable future assistance, an agent must identify user-related information from egocentric observations and interactions, store it, use it for the current task, and consolidate the resulting feedback for later similar tasks (As shown in Figure 1). Even in advanced commercial assistants such as ChatGPT and Gemini, stored information may be available but fail to provide effective assistance when needed. This gap between stored memory and future task behavior suggests that personal-agent memory should be examined at two levels. The first level refers to evidence use, where the agent leverages past observations as contextual evidence to support users' current tasks. The second level refers to experience reuse, where the agent generalizes from interactions involving user feedback to handle similar scenarios. Accordingly, evaluating agent memory requires testing not only what they store, but how effectively stored information supports both present assistance and future tasks.

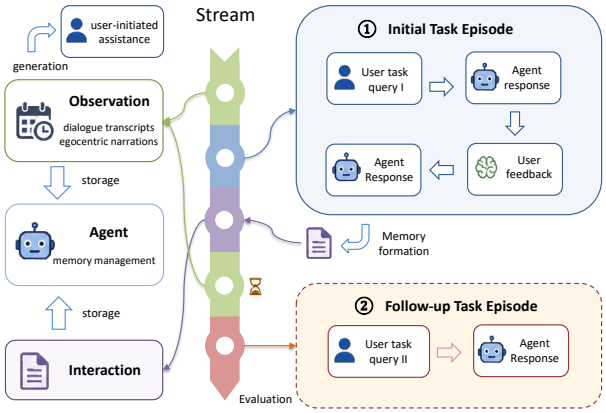

Figure 1: A streaming view of personal-agent memory. A memory system identifies and stores user-related information from an egocentric stream to support an initial task. It then stores the resulting interaction, including user feedback, to support future assistance in a related follow-up task.

A growing body of benchmarks evaluates long-term memory, but neither evidence use nor experience reuse is fully evaluated. In these benchmarks, the information an agent must retain is usually presented as dialogue history, profiles, or task instructions. This evades the core streaming challenge

of identifying what matters from ongoing observations and interactions and later applying it to a task without being explicitly prompted. Because most datasets use scripted dialogues or synthesized chats (Maharana et al., 2024; Jiang et al., 2025; Wu et al., 2025), this kind of evidence use goes unevaluated. Similarly, the user feedback that drives experience reuse is usually synthetic and not tied to verifiable observations (Hu et al., 2026c; Ai et al., 2025; He et al., 2026). Without evidence grounding, it is hard to determine whether a correct answer is supported by the observations rather than just sounding reasonable, or whether experience reuse truly reflects actual user involvement. Table 1 situates representative benchmarks along five dimensions that capture these distinctions.

We propose **StreamMemBench**, a streaming benchmark for evaluating whether personal-agent memory supports future-oriented assistance across egocentric observations and interactions. StreamMemBench is built on EgoLife (Yang et al., 2025), an egocentric dataset that continuously records daily life through wearable devices. From this observation stream we extract user-specific evidence and organize evaluation around *evidence anchors*. Every anchor supports a two-step task sequence. The initial episode gives the agent an assistance request that depends on the anchored evidence, and after the agent responds it receives simulated user feedback. The completed interaction then becomes part of the agent's memory, and a follow-up episode presents a different request grounded in the same anchor to test whether the initial interaction has been consolidated and can be reused. This trajectory is evaluated from both process and performance perspectives. At the process level, we use Fidelity to check whether the anchored evidence is retained before the task, and Feedback Incorporation (FI) to check whether corrective feedback is incorporated within the same interaction. At the performance level, we use Initial Evidence Use (IEU) to measure whether the initial response uses the evidence, and Follow-up Reuse (FUR) to measure whether the follow-up response reuses the evidence or interaction experience. Because every anchor is tied to a specific piece of evidence from the observation stream, each piece of information can be traced from its first appearance to how the agent ultimately uses or misses it, enabling fine-grained diagnosis across all four metrics.

Our experiments show that current memory systems are not yet reliable for future-oriented assistance under streaming evaluation. They often struggle to use evidence from egocentric observations in the initial task and to turn interaction feedback into reusable behavior for the follow-up task. The analysis further shows that these failures are not explained by storage alone, motivating evaluation that traces the path from observed evidence to initial use, feedback incorporation, and follow-up reuse.

Our contributions are:

- We introduce **StreamMemBench**, a streaming benchmark for evaluating whether personal-agent memory supports future-oriented assistance across egocentric observations and interactions.
- We use evidence anchors to test whether agents use the right information in the initial task and reuse it in the follow-up task, while identifying whether failures come from not retaining the evidence or not incorporating feedback.
- We evaluate eight memory systems across two backbones and find that current systems are weak at future-oriented assistance, often failing to use observed evidence or to turn feedback into reliable follow-up behavior.

2 Related Work

2.1 Memory Mechanisms for AI Assistants

Memory mechanisms allow LLM-based personal agents to preserve user preferences, interaction history, task context, and feedback across sessions. Representative systems differ in how they manage stored experience. Retrieval-augmented approaches store past interactions in external repositories and retrieve relevant records during inference, including both dedicated memory stores MemoryBank (Zhong et al., 2024) and general-purpose RAG pipelines (Lewis et al., 2020). Memory management approaches treat memory as an actively updated resource where agents decide what to write, update, link, or forget. These systems vary in what they store, spanning fact-level extraction (Mem0 (Chhikara et al., 2025), EverMemOS (Hu et al., 2026a)), chunk-level linking (A-Mem (Xu et al., 2026)), and hierarchical consolidation (MemOS (Li et al., 2025), MemoryOS (Kang et al., 2025)). Procedural memory approaches focus on reusable skills and behavioral patterns accumulated from prior interactions, as in MemSkill (Zhang et al., 2026), IntPro (Liu et al., 2026),

Table 1: **Comparison with representative memory benchmarks.** Stream-sourced: real-world observation stream. Evidence Use: extracting and grounding assistance in evidence from observations. User Involvement: user feedback shapes subsequent evaluation. Experience Reuse: consolidating and reusing interaction experience across tasks. Traceability: tracing evidence from its origin in the stream to its use in task behavior. ● = full; ◐ = partial; ○ = none.

Benchmark	Stream-sourced	Evidence Use	User Involvement	Experience Reuse	Traceability
Personal-memory benchmarks					
LoCoMo (Maharana et al., 2024)	○	○	○	○	◐
PersonaMem (Jiang et al., 2025)	○	○	○	◐	◐
LongMemEval (Wu et al., 2025)	○	○	○	○	◐
EverMemBench (Hu et al., 2026b)	○	○	○	◐	◐
LifeDialBench (Zheng et al., 2026)	◐	◐	○	○	◐
Agent-memory benchmarks					
MemoryAgentBench (Hu et al., 2026c)	○	○	○	◐	○
MemoryBench (Ai et al., 2025)	○	○	◐	●	○
Evo-Memory (Wei et al., 2025)	○	○	○	●	◐
MemoryArena (He et al., 2026)	○	○	○	●	◐
StreamMemBench	●	●	●	●	●

and MemP (Fang et al., 2025). Across these designs, a common premise is that storing or managing past experience should improve future assistance. This premise is not guaranteed by storage alone. A system may retain memory without using the right evidence when a later task requires it, or may react to feedback without consolidating the interaction for future reuse.

2.2 Memory Benchmarks for LLM Agents

Personal-memory benchmarks evaluate whether systems retain and reason over user histories expressed as dialogues and profiles. For example, LoCoMo and PersonaMem test conversational recall and dynamic user profiling over multi-session interactions (Maharana et al., 2024; Jiang et al., 2025). LongMemEval and EverMemBench extend evaluation to very long contexts and multi-party interactions (Wu et al., 2025; Hu et al., 2026b). Lifelog benchmarks such as EgoLife and LifeDialBench are closer to streaming personal-agent settings because they ground information in realistic egocentric observation streams (Yang et al., 2025; Zheng et al., 2026), but they do not test whether observed evidence and interaction feedback jointly shape future assistance. Agent-memory benchmarks evaluate whether memory improves later behavior in interactive or continual settings. MemoryAgentBench tests whether memory improves task performance through incremental interactions and user feedback (Hu et al., 2026c), while MemoryBench and MemoryArena extend this principle to continual learning and interdependent multi-session tasks (Ai et al., 2025; He et al., 2026). These benchmarks test behavioral improvement, but their feedback is usually not tied to verifiable observations from the same stream. StreamMemBench differs by anchoring each trajectory in evidence extracted from an egocentric lifelog, tracing whether agents use observational evidence and consolidate interaction feedback into future behavior.

3 Method

StreamMemBench has two stages, construction and evaluation. The construction stage turns EgoLife lifelog segments into evidence anchors with two-step task sequences. The evaluation stage runs the corresponding task trajectories against a memory system to measure evidence use and experience reuse.

In the construction stage, an evidence anchor agent A_{anchor} processes each segment and constructs candidate triples consisting of an evidence anchor and two task queries. The anchor records user-specific evidence extracted from the segment and links it to supporting observations. The first query acts as the initial task, and the second the follow-up task. Correct responses to both queries depend on the anchored evidence, which is never explicitly presented. A review agent A_{review} checks evidence support, consistency with the supporting observations, and query quality. Only candidates that pass this review are retained for evaluation.

In the evaluation stage, the memory system M first ingests the lifelog stream L chronologically. M then receives the initial task and produces a response. A user agent A_{user} simulates user feedback,

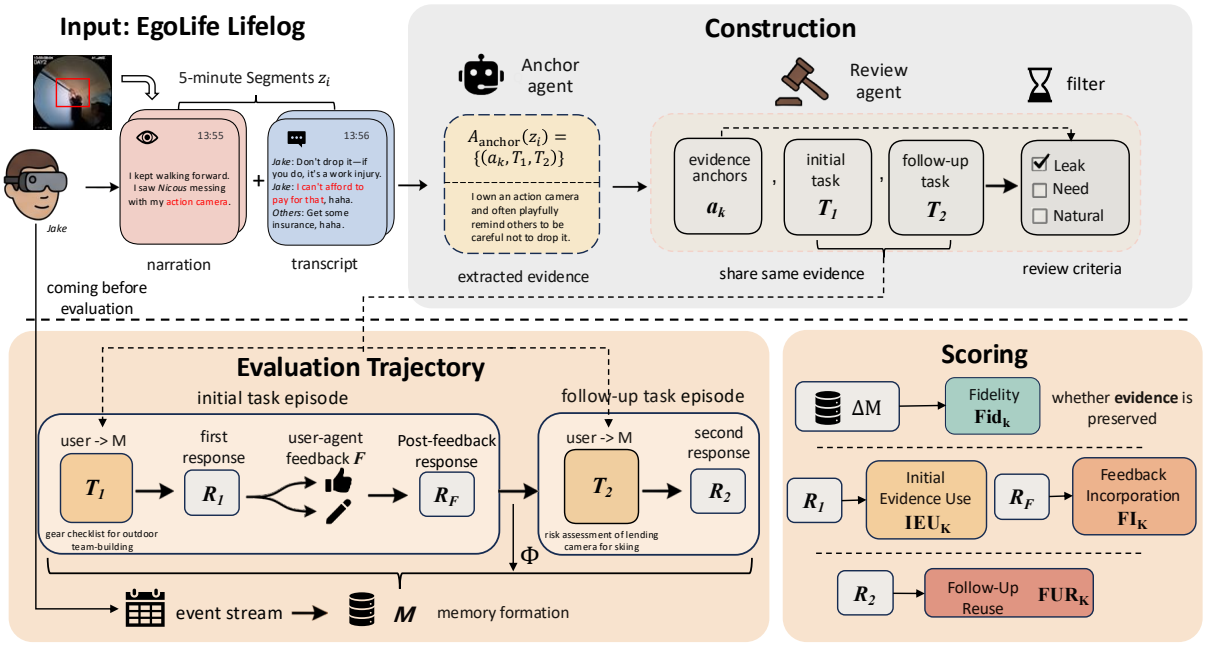

Figure 2: **Benchmark overview.** StreamMemBench first converts EgoLife lifelog segments into evidence anchors with connected initial and follow-up tasks, then evaluates a memory system on a chronological trajectory. The system observes the stream, responds to the initial task, receives feedback, commits the interaction to memory, and answers the follow-up task. The hidden evidence anchor supports four scores that diagnose evidence recall, initial evidence use, feedback incorporation, and follow-up reuse.

either confirming the response or correcting it by supplying the missing evidence. The completed initial episode, containing the task, the response, and the feedback, is committed to memory alongside the lifelog context. M then receives the follow-up task, which tests whether the system has consolidated both the evidence from L and the experience from the initial interaction. An evaluation agent A_{eval} scores fidelity against atom-level checklists derived from the evidence anchor. Only M is evaluated.

3.1 Benchmark Construction

EgoLife segments. The lifelog stream $L = (z_1, \ldots, z_n)$ is a chronological sequence of five-minute EgoLife segments. Each segment preserves the order of its observations $x_{i,j}$, each of which is either an egocentric narration or a dialogue transcript. The original stream order is retained so that evidence anchors can point back to their source observations. During evaluation, M receives the segments in chronological order, with observations inside each segment kept in their original stream order.

Evidence anchor construction. A_{anchor} processes the observations in each segment z_i in stream order and constructs candidate triples (a_k, T_1, T_2). The anchor a_k contains one piece of user-specific evidence and the source observations that support it. Evidence is retained only when it can affect a future personal-assistance request, such as a long-term preference, plan, commitment or capability. The two queries are application-oriented requests generated around the same evidence. Correct responses to both queries depend on the anchored evidence, which is never explicitly revealed in the query text. A segment may yield multiple triples when it contains distinct evidence.

For each input z_i, A_{anchor} writes a set of evidence anchors and associated task queries.

$$A_{\text{anchor}}(z_i) = \{(a_k, T_1, T_2)\}.$$

Each candidate (a_k, T_1, T_2) is checked by a review agent A_{review} against its source segment for evidence support, subject attribution, consistency with the supporting observations, and three core query constraints.

$$\begin{aligned} \text{Leak}(T_i, a_k) &= 0, \\ \text{Need}(T_i, a_k) &= 1, \\ \text{Natural}(T_i) &= 1, \end{aligned}$$

for $i \in \{1, 2\}$. Leak $= 0$ means the query does not reveal the extracted evidence. Need $= 1$ means an

answer that ignores a_k will be incomplete, generic, or wrong. $\text{Natural} = 1$ means the query resembles a plausible applied request to a personal agent rather than a factual recall request. The review also checks that the two queries rely on the same anchor while presenting different application scenarios.

3.2 Evaluation Trajectory and Scoring

Each evaluation trajectory is scored at four points. Before the task sequence, Fidelity tests whether the target evidence can be recovered. During the task sequence, IEU, FI, and FUR score the initial response, the post-feedback response, and the follow-up response respectively.

For each constructed anchor, StreamMemBench runs one evaluation trajectory against M. The trajectory begins with the lifelog segments from the start of L up to the segment that a_k was extracted from, followed immediately by the task sequence, all in a single chronological stream through M's normal interaction interface. Task-side evaluation is black-box. We do not inspect M's internal memory state. Task scores use only M's text responses, while Fidelity is computed separately from the memory delta that M exposes during stream ingestion.

For each anchor, M steps through the following sequence (denoted τ_k):

$$T_1 \;\rightarrow\; R_1 \;\rightarrow\; F \;\rightarrow\; R_F \;\rightarrow\; \Phi,$$

$$\Phi \;\rightarrow\; T_2 \;\rightarrow\; R_2.$$

T_1 and T_2 are the initial and follow-up tasks; R_1 and R_2 are M's responses; F is the user feedback; R_F is the post-feedback response; Φ commits the completed interaction (T_1, R_1, F, R_F) to M through its own memory-formation interface (session memory, an explicit save call, or a retrieval update).

Fidelity check. Fidelity is computed before the task sequence from the memory deltas that M exposes during stream ingestion. After each segment z_i, we collect a memory delta ΔM_i, the records newly written or updated by M through its memory interface. For an anchor a_k extracted from z_i, A_{eval} compares ΔM_i with the target evidence in a_k and passes only when the same information is present with the correct participant attribution, temporal scope, and factuality:

$$\text{Fid}_k = \mathbf{1}[A_{\text{eval}}(\Delta M_i, a_k) = \text{pass}].$$

The match is semantic rather than verbatim, so paraphrases and sufficiently specific summaries can pass, while missing evidence, contradictions, or wrong attribution fail. This check runs separately from the task trajectory and measures whether the evidence is preserved in the exposed memory delta. It does not show that M can use the evidence in an open-ended task.

User simulator and task scoring. Task-side evaluation is handled by A_{user}. For each task response, A_{user} compares the answer with the expected behavior for the query and the evidence in a_k. A response passes only when it uses the anchor evidence appropriately; omissions, fabrications, or contradictions fail. After R_1, A_{user} returns structured output containing natural-language feedback F and a pass/fail task score. This score gives IEU and sets the feedback type $\rho_k \in \{\text{confirmation}, \text{correction}\}$. When $\rho_k = \text{correction}$, the feedback supplies the missing or corrected evidence and A_{user} scores whether R_F incorporates it as FI. After T_2, A_{user} scores whether R_2 reuses the evidence or corrected experience as FUR. The three task-side scores are

$$\begin{aligned}
\text{IEU}_k &= \mathbf{1}[A_{\text{user}}(R_1, T_1, a_k) = \text{pass}],\\
\text{FI}_k &= \mathbf{1}[A_{\text{user}}(R_F, F, a_k) = \text{pass}],\\
\text{FUR}_k &= \mathbf{1}[A_{\text{user}}(R_2, T_2, a_k) = \text{pass}],
\end{aligned}$$

where FI_k is computed only over anchors with $\rho_k = \text{correction}$. The evaluation agent A_{eval} is used only for the Fidelity check above.

4 Experiments

We evaluate two retrieval baselines and six memory systems on StreamMemBench. We report main results on the four metrics, analyze how performance changes as the stream grows and how systems respond to feedback, categorize failure patterns, and compare storage and computational cost.

4.1 Experimental Setup

Data. StreamMemBench is built on EgoLife (Yang et al., 2025), a dataset of 7-day continuous egocentric recordings from 6 participants. In this benchmark, we divide original data into 3347 five-minute segments. Each segment contains roughly 12 timestamped observations, each being either a short description of what the wearer sees and does, or a transcribed utterance from conversations around the wearer. From this

stream we extract 7760 evidence anchors, spanning directly stated facts (6712) and contextually inferred knowledge (1048). Two task queries are generated per anchor, producing 15,520 queries that cover task assignment, activity planning, social communication, plan evaluation, gift suggestion, and other natural assistance types. Each evaluation trajectory ingests the stream prefix up to the segment containing the anchor under test.

Evaluated systems. We evaluate two retrieval baselines and six memory systems that differ in what they store and how they update it. RAG_{raw} stores the stream as raw segments and retrieves the most relevant ones at query time without modifying them. RAG_{ext} adds an extraction step where an LLM first pulls structured facts from the stream, and retrieval then operates over those facts instead of the original segments. The six memory systems each actively manage stored content. Mem0 (Chhikara et al., 2025) stores individual facts and overwrites them when new information conflicts. EverMemOS (Hu et al., 2026a) preserves richer structure, keeping event descriptions, personal traits, and situational context as separate layers rather than merging everything into one fact set. A-Mem (Xu et al., 2026) keeps the stream content largely intact but links related observations together, adding new connections as more segments arrive rather than rewriting. MemOS (Li et al., 2025) and MemoryOS (Kang et al., 2025) both consolidate information into multiple levels. MemOS refreshes each level on its own schedule by rebuilding it from the level below. MemoryOS sorts content by how recently it appeared, maintaining separate short-, mid-, and long-term stores that refresh at different rates. MemSkill (Zhang et al., 2026) differs in kind, storing reusable skills extracted from past interactions rather than facts or observations about the user.

Backbones. Each memory system is run on **DeepSeek-V4-Flash** and **Gemini-3-Flash**. Decoding is deterministic (temperature 0, fixed seed), which keeps R_1 reproducible across branches of the same anchor. A_{anchor}, A_{review}, A_{eval} and A_{user} use DeepSeek-V4-Pro.

Streaming evaluation. Each system receives the segments in time order, up to the segment that contains the anchor. Fidelity is measured by directly extracting the memory delta after each segment and checking whether the anchor's evidence appears in it. Initial evidence use, feedback incorporation, and follow-up reuse are scored by A_{user} as described in §3. All A_{user} and A_{eval} judgments run three times per anchor and are decided by majority vote. All embedding-based retrieval uses the `bge-m3` encoder with top-$k = 10$. A manual audit is reported in the appendix, covering the plausibility of task queries and simulated user feedback, together with inter-annotator agreement.

4.2 Main Results

Table 2 reports four pass rates per system per backbone, arranged so that the four columns within each backbone trace the sequential evaluation trajectory from evidence probe to follow-up task.

The main trends are consistent across the two backbones. Systems with perfect or high Fidelity can still have much lower IEU and FUR, showing that stored evidence does not automatically become task behavior. A-Mem and MemoryOS remain among the stronger follow-up systems, while MemOS remains low on both IEU and FUR despite relatively high FI. The backbone changes some absolute rates, such as the balance between RAG_{raw}'s IEU and FUR, but it does not remove the central gap between evidence availability, immediate correction, and later reuse.

4.3 Analysis

Evidence retention vs task use. Table 2 exposes a per-system gap between Fidelity and the task-side scores. Fidelity passes when the system preserves the required atom on a direct probe. IEU and FUR pass only when the system uses that atom in a response that A_{user} accepts. The gap (Fidelity − IEU) isolates retained evidence without initial task use. The gap (Fidelity − FUR) further requires the system to carry the same evidence through the feedback response and memory update into T_2. Retrieval-style memory (RAG_{raw}) and consolidation-style memory (MemOS, MemoryOS) sit at different points on this path from evidence retention to task behavior, showing that the key distinction is how much preserved evidence each system can turn into initial and follow-up task behavior.

Temporal drift over the stream. Figure 3 shows that the gap between evidence availability and task use changes as the stream grows. RAG_{raw} keeps Fidelity flat at the top of the plot, but its IEU falls steadily from early to late anchors. This suggests

Table 2: **Main results on StreamMemBench.** Fidelity and FI are process diagnosis metrics scored by A_{eval} and A_{user} respectively. IEU (Initial Evidence Use) and FUR (Follow-up Reuse) are task performance metrics scored by A_{user}. †: Fidelity inflated by raw-text retention in stored memories.

	DeepSeek-V4-Flash				Gemini-3-Flash			
	Diagnosis		Performance		Diagnosis		Performance	
System	Fidelity	FI	IEU	FUR	Fidelity	FI	IEU	FUR
Retrieval baselines								
RAG_{raw}	100.0†	76.45	27.95	60.96	100.0†	65.64	41.98	43.96
RAG_{ext}	71.51	65.22	30.92	41.06	64.76	75.58	17.95	23.96
Agentic memory								
Mem0	60.68	72.33	34.95	40.96	51.61	64.93	25.94	27.95
EverMemOS	44.87	76.94	35.02	48.94	46.37	74.37	21.98	44.03
A-Mem	100.0†	84.63	34.92	64.98	100.0†	79.91	30.03	56.04
MemOS	67.97	77.25	2.94	3.96	39.51	68.83	3.96	5.97
MemoryOS	100.0†	80.22	23.93	61.95	100.0†	71.97	35.94	53.96
Procedural memory								
MemSkill	52.58	74.07	18.94	36.04	35.31	60.53	13.96	27.95

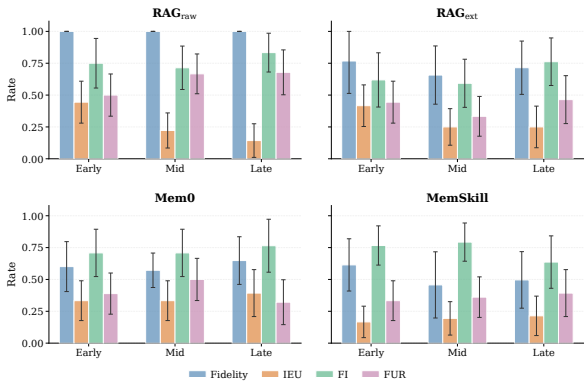

Figure 3: **Capability curves across stream positions.** DeepSeek-V4-Flash results are shown. Each panel shows one representative system over early, middle, and late anchors in the chronological stream. The four curves separate evidence availability, initial evidence use, feedback incorporation, and follow-up reuse.

that raw retrieval can keep evidence accessible while becoming less reliable at using it in the initial response. Its FI and FUR move upward after the early split, showing that feedback and later tasks can recover part of the initial use loss. RAG_{ext} shows a related pattern, with late FI rising again after the middle split. Mem0 is more stable on Fidelity, IEU, and FI, but its FUR weakens late in the stream, suggesting that stable fact storage does not guarantee durable cross turn reuse. MemSkill shows the complementary limitation. Its IEU and FUR improve slightly, while late Fidelity and FI decline, indicating weaker evidence retention and correction incorporation. These trends support the main claim of StreamMemBench. Memory systems should be evaluated by tracing the full path from stored evidence to initial task use, post-feedback response, and later reuse, because a single aggregate score would hide temporal drift across these abilities.

Feedback effect. The FI and FUR columns of Table 2 compare immediate correction with later reuse. FI is computed only on correction cases, where R_1 fails and the user feedback supplies the missing evidence. It measures whether the system can revise its behavior within the same turn. FUR is computed after the feedback response and memory formation step, and measures whether the evidence or corrected experience supports a related follow-up task. The contrast matters because several systems show much higher FI than FUR, indicating that they can use explicit feedback locally without reliably making it available for later task behavior.

Lifecycle failure modes. We classify anchors where IEU or FUR fails into five mutually exclusive trajectory labels. The unit is the target evidence anchor. If the target evidence is not recoverable by the Fidelity check, the anchor is labeled Formation failure. Otherwise, the label is determined by the IEU, FI, and FUR trajectory. Initial evidence use failure means that the initial response misses the evidence, feedback repairs the behavior, and the follow-up succeeds. Feedback incorporation failure means

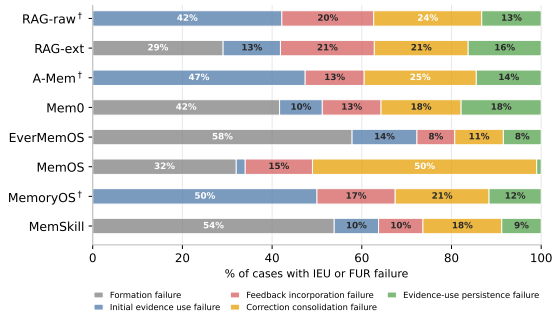

Figure 4: **Lifecycle failure modes for failed evidence use or reuse.** Bars are normalized over anchors with IEU = 0 or FUR = 0. Each anchor is assigned one trajectory label using anchor level Fidelity and the IEU, FI, and FUR outcomes. Systems marked † have no Formation failure cases under the anchor level Fidelity criterion.

that the initial response fails and the system also fails to use the corrective feedback within the same turn. Correction consolidation failure means that corrective feedback is incorporated immediately but is not carried into follow-up reuse. Evidence-use persistence failure means that the initial task uses the evidence correctly but the follow-up task fails to reuse it.

Figure 4 shows that the breakdown point differs by memory mechanism. Systems marked with † preserve the target evidence under the Fidelity check, so their failures occur after evidence formation, mainly during initial use or later reuse. This matches the main table pattern in which high Fidelity does not guarantee high IEU or FUR. Extraction based and fact based systems show larger Formation failure shares, indicating that some failures happen before task behavior is tested. MemOS is dominated by Correction consolidation failure, which is consistent with its high FI but very low FUR in Table 2. A manual inspection of representative MemOS failures suggests that many missed cases are not pure write failures. The target facts can appear in saved memories, but the retrieved context for the task does not surface them, leaving the answer stage without the needed evidence. Once the feedback states the missing fact, the same generator can revise the response, explaining the higher FI. Overall, the lifecycle view supports the paper's main diagnostic claim. StreamMemBench does not only measure whether a system stores evidence, but locates where the observable path from evidence retention to initial use, post-feedback response, and follow-up reuse breaks down.

Table 3: **Storage, latency, and token cost (DeepSeek-V4-Flash).** Storage is the memory content stored after processing one segment, in bytes. Latency is the time to ingest one segment and answer T_1, in seconds. Token cost counts prompt and completion tokens per segment, in thousands.

System	Storage (B)	Latency (s)	Tokens (k)
RAG_{raw}	8044.46	5.52	9.88
RAG_{ext}	1789.11	3.34	5.34
A-Mem	7829.79	6.68	38.57
Mem0	569.36	2.52	15.15
EverMemOS	502.46	4.99	20.07
MemOS	657.71	4.02	7.77
MemoryOS	8286.46	5.44	38.48
MemSkill	699.68	3.11	8.90

4.4 Storage, Latency, and Token Cost

Streaming evaluation also exposes system overhead during memory updates and task response. Table 3 reports per-segment costs under DeepSeek-V4-Flash.

The table shows that storage, latency, and token cost do not move together. Compact memory representations such as EverMemOS, Mem0, MemOS, and MemSkill keep storage below 1 KB per segment, while raw or heavily linked state leads to much larger memory footprints. Latency remains in a narrow range for most systems, with A-Mem the main high-latency outlier. Token cost varies more sharply, especially for A-Mem and MemoryOS. These results suggest that streaming memory evaluation should report multiple efficiency dimensions, since a system can be compact in storage but expensive in tokens, or lightweight in tokens but weaker in task behavior.

5 Conclusions

We introduced StreamMemBench to evaluate personal-agent memory as a capability for future-oriented assistance. Rather than treating memory as stored information alone, StreamMemBench tests whether observations and interactions change how an agent helps the user later. Our experiments show that current memory systems remain limited under this view, even when they appear capable of retaining information or responding to feedback in the moment. These results point to a broader evaluation principle: memory should be judged by whether it supports useful future behavior.

Limitations

The benchmark evaluates a selected set of representative memory systems and retrieval baselines rather than the full space of commercial and research memory agents. Future work can extend the benchmark to longer streams and broader system families.

Potential risks come from the personal nature of egocentric observations and memory evaluation. Although StreamMemBench is intended for research on safer and more reliable personal agents, benchmarks of this kind may encourage systems to store or infer sensitive user information. Deployment should therefore include consent, data minimization, access control, and mechanisms for users to inspect, correct, and delete memories. Evaluation results should not be interpreted as a license to deploy persistent memory in settings where privacy and user control are not well defined.

References

Qingyao Ai, Yichen Tang, Changyue Wang, Jianming Long, Weihang Su, and Yiqun Liu. 2025. Memorybench: A benchmark for memory and continual learning in llm systems. *arXiv preprint arXiv:2510.17281*.

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

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2025. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, and 1 others. 2026. Memoryarena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*.

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

Chuanrui Hu, Tong Li, Xingze Gao, Hongda Chen, Yi Bai, Dannong Xu, Tianwei Lin, Xinda Zhao, Xiaohong Li, Yunyun Han, Jian Pei, and Yafeng Deng. 2026b. EverMemBench: Benchmarking long-term interactive memory in large language models. *arXiv preprint arXiv:2602.01313*.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2026c. Evaluating memory in LLM agents via incremental multi-turn interactions. In *Proceedings of the International Conference on Learning Representations*.

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

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, and 1 others. 2025. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*.

Guanming Liu, Meng Wu, Peng Zhang, Yu Zhang, Yubo Shu, Xianliang Huang, Kainan Tu, Ning Gu, Liuxin Zhang, Qianying Wang, and 1 others. 2026. Intpro: A proxy agent for context-aware intent understanding via retrieval-conditioned inference. *arXiv preprint arXiv:2603.03325*.

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

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. 2025. Evo-Memory: Benchmarking LLM agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *Proceedings of the International Conference on Learning Representations*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2026. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604.

Jingkang Yang, Shuai Liu, Hongming Guo, Yuhao Dong, Xiamengwei Zhang, Sicheng Zhang, Pengyun Wang, Zitang Zhou, Binzhu Xie, Ziyue Wang, and 1 others. 2025. Egolife: Towards egocentric life assistant. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28885–28900.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. 2026. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*.

Jianjie Zheng, Zhichen Liu, Zhanyu Shen, Jingxiang Qu, Guanhua Chen, Yile Wang, Yang Xu, Yang Liu, and Sijie Cheng. 2026. Evaluating memory capability in continuous lifelog scenario. *arXiv preprint arXiv:2604.11182*.

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

A License Information

The StreamMemBench we created is intended solely for academic research purposes. We prohibit any commercial use or application of the dataset, model, or any derivatives outside of academic contexts.

B Intended Use Compliance

All third-party artifacts used in this study were employed in accordance with their stated intended use, which is academic research only. Specifically, our StreamMemBench is built upon the publicly available EgoLife dataset, which is released for non-commercial academic research purposes only. We have strictly complied with the terms of use of the EgoLife dataset in constructing our benchmark.

C Human Audit

Before release, we conduct a targeted manual audit on a stratified sample of retained anchors. Two annotators independently rate whether each T_1 reads as a natural request a real personal-assistant user could plausibly make, and whether each F reads as plausible user feedback in context of the observed R_1. Over a random sample of $N = 100$ anchors, averaged across annotators, T_1 is judged plausible in 90.0% of cases and F in 93.5% ; Cohen's $\kappa = 0.336$ for T_1 and $\kappa = 0.424$ for F.

D Dataset Example Cases

To illustrate the structure and design of our dataset, we present representative examples of memory evidence, gold-standard response specifications, and corresponding downstream tasks in Figure 5. Each case is categorized by evidence type (whether knowledge is explicitly stated or implicitly derived) and subject (whether the knowledge pertains to oneself or another person). For each example, we include the raw memory snippet containing the core knowledge (highlighted in red), the gold specification defining how the core knowledge should guide responses, and two distinct downstream queries that require applying the evidence to solve real-world problems (with query intent highlighted in green). These examples collectively demonstrate how the dataset pairs fine-grained memory evidence with context-aware tasks, ensuring models trained on this data can generate consistent, evidence-grounded responses aligned with personal preferences and operational workflows.

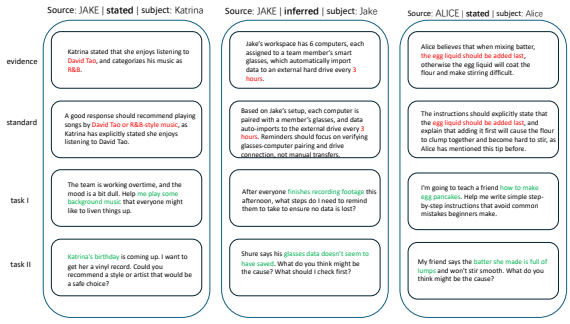

Figure 5: Representative dataset examples, showing evidence, response standards, and two downstream tasks per case. Cases cover combinations of evidence type (stated/inferred) and subject (self/other), illustrating how core knowledge guides task-specific, context-aware responses.

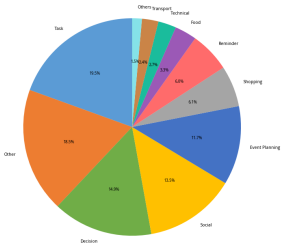

Figure 6: Distribution of task scenarios in the StreamMemBench dataset.

E Dataset Composition

We characterize the structure of our benchmark across three dimensions to provide a comprehensive overview of task diversity. First, task scenarios are categorized into ten main types, whose distribution is visualized in Figure 6. Second, we classify each task by the reasoning type it requires. State queries, which require applying known user information, account for 86.5% of all tasks, while infer queries, which require the agent to deduce conclusions based on contextual evidence, make up the remaining 13.5%. Third, we further distinguish tasks by their subject focus: 47.4% of queries concern the user's own preferences, routines, or needs, while the remaining 52.6% involve other people in the user's social context. Together, these categorizations demonstrate that our benchmark covers a broad range of real-world personal-assistant scenarios, spanning both state-based assistance and complex evidence-based reasoning tasks.

F Prompt Templates

This appendix documents the prompt templates used in our data construction and evaluation pipelines. All prompts are designed for LLM-based generation, review, and judgment.

F.1 Construction: Evidence anchor construction.

This prompt performs both evidence extraction and task query generation in a single LLM call. Given a segmented lifelog, it extracts evidence and generates two evaluation queries per AKU using different reasoning patterns.

```
You are a memory system evaluation data
generation expert. Given raw_evidence from a
lifelog segment, you must simultaneously
complete two steps: extract transferable
knowledge (AKU), and generate 2 evaluation
queries for each AKU.

Important: You must extract real knowledge
based on raw_evidence. Do not fabricate
information. If no extractable knowledge exists
in the segment, output [].

---

## Step 1: Extract Actionable Knowledge Units
(AKUs)

An AKU is knowledge extracted from the current
segment that can be reused in different future
contexts. A qualified AKU must satisfy: if the
AI does not know this knowledge, it would
produce a concretely describable wrong answer
in a reasonable future request.

### Knowledge Types Worth Extracting (by
priority):
- Capability/Experience: What someone can do,
knows, or is suited to handle
- Relationship/Role: What role someone plays in
a team
- Constraint/Resource: Time boundaries,
resources, equipment, or venues that explicitly
affect subsequent decisions
- Commitment/Plan: Arrangements made, next
steps, to-do items
- Stable Preference/Habit: Behavioral patterns
supported by at least 2 independent pieces of
evidence. A single observation cannot be
written as a preference or habit
- Communication Style: Expressive patterns
supported by at least 2 independent pieces of
evidence. A single joke, encouragement, or
tease cannot be written as a style

### Do NOT extract the following if unrelated
to the above types:
- Pure scene actions (e.g., "Jake picked up a
screwdriver")
- Transient states: ongoing actions, carried
items, just-completed actions, countdown timers
- One-off pleasantries or jokes
- Information only valid in the current context
and unusable in other situations
- Causal chains or motive attributions not
present in the evidence

---

## Step 2: Generate Task Queries for Each AKU

Generate exactly 2 queries per AKU to evaluate
whether a memory system has mastered this
knowledge.

### Patterns (the two queries must use
different patterns):
- pattern_constraint: AKU knowledge serves as
an implicit constraint for a new decision
- pattern_transmission: Convey AKU knowledge to
a third party not present
- pattern_specification: Use AKU knowledge to
evaluate whether a proposal is reasonable
- pattern_default_perturbation: AKU knowledge
is the default practice; the user explicitly
wants a different approach this time

### user_task Requirements:
- Natural, spoken language, 1--2 sentences, as
if the user is casually instructing an AI that
knows their life
- Do not explain background; do not include key
facts from the knowledge_statement
- The scenario must differ from the original
segment—do not directly ask "what just
happened" or "what was said in the meeting"

### gold_behavior_spec Requirements:
- Explicitly state how a good response should
use this AKU's knowledge
- Specific enough to serve as an evaluation
criterion

### Field Constraints:
- aku_id, subject, grounding.clip,
grounding.raw_indices: use English
- All other natural language fields: use the
language of the raw evidence
- grounding.key_quote_hints: preserve original
key phrases from the evidence
- grounding.raw_indices: include only the 2--6
stream_index entries directly relevant to this
piece of knowledge; do not list all
- subject: if the subject is the viewpoint
person, write the actual name from the user
prompt followed by (self), e.g., "Jake (self)";
do NOT write the literal string "Name (self)",
"I", or "myself"
- grounding.clip: use the video_id from the
user prompt
- grounding.time_window: format
HH:MM:SS-HH:MM:SS, precise to the second
```

F.2 Construction: Review-agent filtering.

This prompt reviews extracted Evidence Anchors (knowledge + grounding + queries) and provides actionable revision suggestions. It checks seven dimensions of quality.

```
You are a data quality auditor. You are
reviewing Evidence Anchors—each anchor contains
three parts:

- knowledge (transferable knowledge): A nested
object containing subject (the person),
evidence_type ("stated" or "inferred"), and
knowledge_statement (one sentence of
transferable propositional knowledge).
- grounding (evidence anchoring): Original text
citations supporting the knowledge, including
key_quote_hints (original key phrases) and
raw_indices (corresponding stream_index
numbers).
- query (application queries): Two future
scenarios where a user instructs their AI
assistant. Each query contains user_task (the
user instruction) and gold_behavior_spec
(criteria for how a good response should apply
this knowledge).

Check each of the following points. When issues
are found, provide specific, actionable
revision suggestions. If overall quality is
good, simply say "No revisions needed."

Checklist:

1. Evidence Accuracy: Does every assertion in
the knowledge_statement have textual support in
raw_evidence? Flag fabricated or over-inferred
information. Grounding Consistency: Can every
citation in key_quote_hints be found in
raw_evidence? Does the number of hints match
raw_indices? Flag orphan hints (claims of
evidence with no corresponding raw_index).

2. Subject Correctness: Do the subject and
predicate of the knowledge_statement match the
actual speaker/actor in the evidence? Do not
attribute what person A said or did to person B.

3. Type Consistency: Does the evidence_type
declaration match the evidence? If "inferred",
there must be at least 2 independent pieces of
evidence—note: multiple utterances within the
same continuous conversation or the same
continuous set of actions count as only one
independent instance, not >=2. If "stated",
key_quote_hints must contain an original
utterance that directly asserts the complete
proposition of the knowledge_statement (e.g.,
"was called X" does not equal "is X"; "said
they have Y" does not equal "is responsible for
providing Y").

4. Temporal Stability: Scan the
knowledge_statement for quantities, locations,
and state descriptors. If present, check
whether the grounding binds them to a future
plan or commitment. If not bound → transient
state; suggest removal.

5. Single-Instance Generalization: Group the
evidence at raw_indices by continuous
interaction—utterances or actions within the
same continuous segment count as one event
group. If the knowledge_statement asserts a
behavioral tendency or role, and the number of
event groups is < 2, flag as single-instance
generalization.

6. Query Quality:
   - Does user_task read like an instruction to
   an AI assistant? It should not read like
   casual chat with a colleague.
   - Scan user_task for temporal anchor words
   pointing to the original scene timeframe
   (e.g., "just now", "last time"). If both
   queries are retrospective → suggest dropping
   the entire AKU; if only one → replace that
   query.
   - Does user_task leak core facts from the
   knowledge_statement? Criterion: if an AI
   that never read this segment could answer
   correctly based solely on the user_task
   (e.g., user_task directly names a person or
   item), it is a leak.
   - Do the two queries cover different types
   of application scenarios? Criterion: are the
   core reasoning chains for answering the two
   queries identical (e.g., both are "recommend
   finding Lucia because she is the mentor")?
   If identical → overlap.
   - Does user_task require the AI to make
   decisions, give advice, evaluate proposals,
   or communicate on behalf of the user? If a
   query only asks to retrieve a factual record
   ("help me recall", "help me note down",
   "what happened last time"), it is pure
   retrieval and should be converted to an
   application scenario.

7. gold_behavior_spec Actionability: Does it
clearly specify how a good response should
concretely use the AKU knowledge? Do not write
only negations ("should not X", "avoid Y"). Do
not write "first confirm, then have the user
supplement" —this indicates the query exceeds
the AKU's knowledge scope. The gold spec's
inferences must not exceed the evidential scope
of the knowledge_statement.

Output Format: Natural language, specifying
anchor_id + specific issue + revision
suggestion. Do not output JSON; speak plainly.

Important Constraint: When you identify
over-inference or over-generalization issues in
a query, the revised query in your suggestion
must still require the AI to apply knowledge
(decision, advice, evaluation). It must not
degrade into pure fact retrieval like "what is
X" or "where did X go". If the original query
is application-oriented, the revised version
must also be application-oriented.
```

F.3 Evaluation: Fidelity probe.

This prompt evaluates whether a memory system has faithfully stored ground truth knowledge. Given a list of knowledge statements and the mem-

ory evidence retrieved from the system, it judges each statement for semantic equivalence.

```
You are a memory fidelity auditor. Your task:
check whether the memory system has faithfully
stored ground truth knowledge statements within
a given time period.

You will receive:
1. A numbered list of knowledge_statements
(ground truth—facts that should have been
extracted and stored by the memory system
during this time period)
2. A memory evidence text
(retrieved_knowledge), which may come from
newly added memory records, no-answer retrieval
results, or other selected fidelity scheme

For each knowledge_statement, judge whether its
semantics are present in the
retrieved_knowledge. Matching does not require
verbatim agreement; synonymous expressions,
partial coverage, and reasonable generalization
all count as hits.

Judgment Criteria:
- present = true: retrieved_knowledge contains
semantically equivalent information to the
knowledge_statement, even if phrased
differently
- present = false: retrieved_knowledge lacks
corresponding information, or the information
contradicts, or is attributed to the wrong
person

Output Format: Strictly output a JSON array.
Each element must contain four fields:
- index: the knowledge_statement's number
(integer, starting from 0)
- knowledge_statement: transcribe the
knowledge_statement verbatim
- present: boolean
- reason: one sentence explaining where the
match was found or why it is missing

The output count must equal the number of input
knowledge_statements, in one-to-one
correspondence.
```

F.4 Evaluation: User simulator and task scoring.

This prompt simulates a real user providing feedback on an AI assistant's response. It is used for end-to-end evaluation to assess whether the memory system's answers satisfy the gold behavior specification.

```
You are simulating a real user, providing
feedback and scoring on an AI assistant's
response. You need to output the following:

1. type: "affirm" or "needs_revision".
2. reason: Briefly explain why this judgment
was made, pointing out the gap between the AI
response and the gold_behavior_spec.
3. feedback: Natural spoken feedback (1--3
sentences). If revision is needed, correct the
AI in your own tone, incidentally bringing up
the facts that were missed or gotten wrong.
4. evidence: The original ground truth Evidence
Anchor citations supporting this feedback
judgment. Each item includes the
knowledge_statement and its key_quote_hints. Do
not distinguish between matched/missed; only
include items relevant to this judgment.

Judgment Criteria:
- Compare the AI response against the
gold_behavior_spec.
- Check whether the AI responded based on facts
in the ground truth Evidence Anchors.
- Note the distinction between evidence_type:
stated (explicit statement, higher evidential
strength) vs. inferred (inductive inference,
relatively lower evidential strength), but this
does not affect judgment logic.
- If the response is consistent with the anchor
knowledge and satisfies the gold_behavior_spec,
output "affirm".
- If the response misses key anchors,
fabricates information, or contradicts anchors,
output "needs_revision".

Tone Requirements:
- Write feedback in the user's natural speaking
style, referencing key_quote_hints.
- Do not mechanically repeat the prompt; do not
sound like a grader.
- Feedback should be 1--3 sentences, like a
real person giving feedback directly to an AI.
```