MEMFUSE: MULTI-SOURCE MEMORY FUSION FROM FRAGMENTED OBSERVATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term memory is essential for agents that operate across extended interactions, yet existing memory systems and benchmarks predominantly focus on single-source textual histories. In realistic settings, however, relevant information is often fragmented across applications and devices, as well as across users and time, requiring agents to integrate dispersed observations into coherent episodic memories while preserving their source provenance. To address these gaps, we introduce **MemFuseBench**, a benchmark for *multi-source memory fusion*. MemFuseBench is built with a Scene-to-Sensor pipeline that synthesizes controllable scenarios into source-tagged observations, evidence-grounded questions, and adversarial distractors. It enables systematic evaluation of temporal reasoning, cross-source evidence fusion, and robustness to noise. We further propose **MemFuse**, a structured memory system that preserves source-level evidence in event-layer atomic memory and organizes related atomic events into cluster-layer fused memory within a causal fusion graph. During retrieval, MemFuse retrieves and organizes related evidence fragments while maintaining traceability to original source events. Experiments on MemFuseBench show that MemFuse achieves the best overall performance among the evaluated memory systems under all three LLM settings and consistently improves performance on questions requiring cross-source evidence fusion.

1 INTRODUCTION

Long-term memory systems have received growing attention as agents are expected to maintain useful context across extended interactions. Existing systems store past interactions as records, summaries, or structured memories for retrieval (Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025; Hu et al., 2025; 2026a; Cao et al., 2026; Sun et al., 2026; Hu et al., 2026d), and work well when relevant context forms a coherent history. As user expectations grow, however, agents need to remember not only what users explicitly tell them, but also useful observations from devices, applications, and other users. In this setting, the same underlying episode may then be represented by fragmented events from different origins, making it necessary to integrate complementary observations without losing their sources. We call this memory-level problem *multi-source memory fusion*: retrieving and integrating distributed semantic events while preserving event-layer provenance.

Existing memory benchmarks mainly evaluate conversational recall, temporal updates, or long-context reasoning over interaction histories (Maharana et al., 2024; Wu et al., 2025; Tan et al., 2025; Hu et al., 2026b; Yang et al., 2026b). Recent benchmarks consider heterogeneous digital traces and multimodal evidence (Cheng et al., 2026; Chai et al., 2026), but they do not specifically test whether memory systems can link fragmented, source-tagged events into traceable evidence for fusion-oriented questions. It therefore remains difficult to test whether a system can recover complementary observations from different origins.

To address these challenges, we first introduce **MemFuseBench**, a benchmark for multi-source memory fusion. Starting from controllable scenarios, its Scene-to-Sensor pipeline generates source-tagged observations, evidence-grounded questions, and adversarial distractors. Figure 1 illustrates a question requiring evidence from multiple origins while ignoring a plausible distractor. The benchmark contains 357 questions over 7,823 events across six diagnostic categories.

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

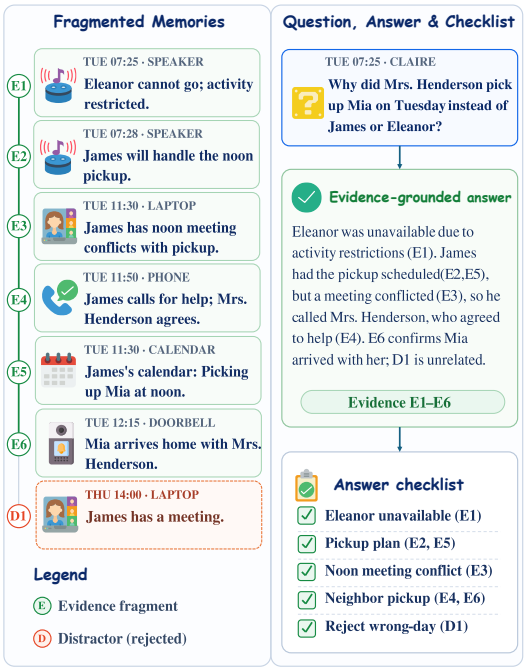

Figure 1: A MemFuseBench instance requiring fusion across a household conversation.

To further support multi-source memory fusion, we propose **MemFuse**, a graph-structured memory system that preserves event-layer memory as the evidence layer and organizes related events into cluster-layer memory within a causal fusion graph. At retrieval time, MemFuse uses agentic search to recover both compact cluster memories and their traceable source events.

In summary, our contributions are: (1) identifying multi-source memory fusion as a memory-level research problem over fragmented, source-tagged events; (2) introducing MemFuseBench, a fusion-centric benchmark for this setting, featuring six diagnostic categories and answer checklists, supported by a scalable LLM-based synthesis and reviewer-corrector validation pipeline; (3) proposing MemFuse, a structured memory system that preserves source-level evidence in event-layer atomic memory and fuses related events into cluster-layer memory within a causal fusion graph; and (4) validating MemFuseBench and MemFuse through experiments across multiple models, where MemFuse achieves the best Overall score among the evaluated retrieval and memory systems under all three LLM settings.

2 Related Work

Long-Term Agent Memory. Agent-memory systems store past interactions as records, summaries, linked notes, or hierarchical memory units for later retrieval (Park et al., 2023; Packer et al., 2023; Zhong et al., 2024; Liu et al., 2023; Wang et al., 2023; Chhikara et al., 2025; Xu et al., 2025; Zhang et al., 2025; Hu et al., 2025; 2026a; Cao et al., 2026; Sun et al., 2026). Recent work also introduces structured or graph-based memory to connect related experiences and support multi-step retrieval (Hu et al., 2026c; Xu et al., 2026; Yang et al., 2026a; Hu et al., 2026d). These systems improve organization over flat logs, but they mainly assume conversational or interaction histories rather than fragmented observations across different sources.

Memory Benchmarks. LoCoMo, LongMemEval, and MemBench evaluate conversational recall, temporal updates, abstention, and long-context reasoning (Maharana et al., 2024; Wu et al., 2025; Tan et al., 2025). EverMemBench and GroupMemBench extend this line to long-term interactive memory and multi-party conversations (Hu et al., 2026b; Yang et al., 2026b). LifeBench and SMM-Bench move closer to realistic settings by using heterogeneous digital traces or independently originated multimodal evidence (Cheng et al., 2026; Chai et al., 2026). They broaden the input sources

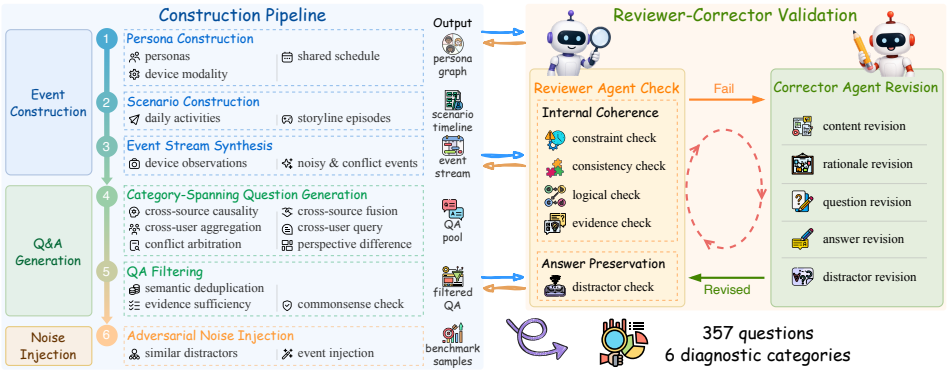

Figure 2: Overall MemFuseBench construction framework. The left panel summarizes the **construction pipeline**, whose six stages are organized into three phases: event construction, QA generation, and noise injection. The right panel shows **reviewer-corrector validation**, where candidate samples are checked, revised, and iterated until the final accepted sample is produced.

for memory evaluation, but do not center the task design on linking, fusing, and tracing distributed event fragments.

Multi-Source memory Fusion. Event-centric resources and lifelogging systems organize heterogeneous records into event structures for temporal or commonsense reasoning (Gottschalk & Demidova, 2018; Zhang et al., 2020; Gurrin et al., 2014). Multisensor fusion studies how signals or decisions from multiple sensors are registered, estimated, or combined (Khaleghi et al., 2013). These research directions are related to multi-source evidence, but they are not concerned with agent memory: none addresses how a long-term memory system should preserve atomic provenance while grouping and retrieving fragmented observations for downstream question answering.

3 MemFuseBench

Most memory benchmarks evaluate recall within a single conversational history. Recent benchmarks extend evaluation to heterogeneous traces and cross-source multimodal reasoning; we focus on controlled, source-tagged atomic event streams with auditable evidence and distractors. To address these challenges, we introduce MemFuseBench.

3.1 Dataset Construction

Each MemFuseBench instance contains source-specific event streams and a question with its reference answer and an answer checklist. We construct these instances with a Scene-to-Sensor pipeline that turns controllable latent scenarios into source-specific observations, evidence-grounded questions, and adversarial non-evidence events. Reviewer-corrector validation checks and repairs intermediate artifacts for consistency, evidence support, and answer preservation under injected distractors.

Construction Pipeline. The core construction method is **Scene-to-Sensor** generation, which builds each instance by top-down narrowing rather than directly generating question-answer pairs. It first fixes a high-level ground truth—stable personas and a causally linked scenario timeline (*what happens in the world*)—then projects each latent activity into concrete, timestamped, source-specific event streams (*what each device observes*), from which the evidence-grounded questions are derived. Figure 2 summarizes the framework, which is organized into six stages:

1. **Persona Construction.** Sample stable personas and their shared context—schedules, relationships, and multi-device inventories with observable modalities—into a persona-source graph, which constrains all later stages.

Category	#Q	#E	Description
Cross-source causality (Causal)	63	8.5	Infer causal relations across source observations.
Cross-source fusion (Fusion)	71	15.2	Combine partial observations across sources.
Cross-user aggregation (User Agg.)	52	11.4	Aggregate evidence across users.
Cross-user query (User Query)	61	7.1	Answer queries about another user's events.
Conflict arbitration (Conflict)	70	2.8	Resolve conflicting source reports.
Perspective difference (Perspective)	40	13.3	Account for different user or source perspectives.
Total	357	9.4	Full benchmark release.

Table 1: Per-category statistics of the current MemFuseBench release. #Q is the number of questions; #E is the average number of distinct evidence events required per question.

2. **Scenario Construction.** Organize the personas' daily activities into causally linked storyline events grouped by episode. This timeline is the latent ground truth, later refracted into source-specific events.

3. **Event Stream Synthesis.** Render the scenario into source-specific events, interleaving routine, periodic, noise, and conflict events so each source gives only a partial view. This yields a timestamped event stream.

4. **Category-Spanning Question Generation.** Derive questions spanning the six diagnostic categories in Table 1, producing a QA pool of questions, reference answers, answer checklists.

5. **QA Filtering.** Filter the pool to remove semantic duplicates, commonsense shortcuts, and items whose labeled evidence cannot support the reference answer, retaining only answerable, evidence-supported, non-duplicative items.

6. **Adversarial Noise Injection.** Inject semantically similar distractors to complicate retrieval while keeping the gold evidence set auditable and unchanged, so finalized samples contain both gold evidence and plausible non-evidence distractors.

Validation Protocol. As the pipeline builds each instance stage by stage in a top-down manner, an inconsistent persona-source graph can propagate errors to every later artifact. To prevent this, MemFuseBench applies **reviewer-corrector validation** to every intermediate artifact before it enters the next stage: a *reviewer* flags structural, semantic, and consistency issues with stage-specific prompts, a *corrector* repairs them, and this review-correction process iterates until the reviewer finds no remaining issues.

The criteria are stage-specific and focus on two targets. For the first four stages, validation targets *internal coherence*: the persona-source graph, scenario timeline, event stream, and QA pool must each be self-consistent and aligned with the artifacts upstream. For Adversarial Noise Injection, validation instead targets *answer preservation*: injected distractors must complicate retrieval without altering the gold evidence set or reference answers.

We further conduct a model-guided verification pass: given the full context, GPT-5.5, Claude Opus 4.6, and Gemini 3.1 Pro each independently answer and evaluate every question, and the samples on which they disagree are manually inspected and corrected. This procedure results in at least one revision for approximately 20% of the 357 QA instances.

3.2 Dataset Analysis

This pipeline yields MemFuseBench with six scenario datasets, each paired with evidence-grounded questions spanning six diagnostic categories. On average, a scenario contains 1,303.8 atomic events and 110.6k tokens under the Gemini 3.1 Flash Lite tokenizer. Table 1 summarizes the resulting 357 questions; each question requires evidence from 9.4 distinct events on average.

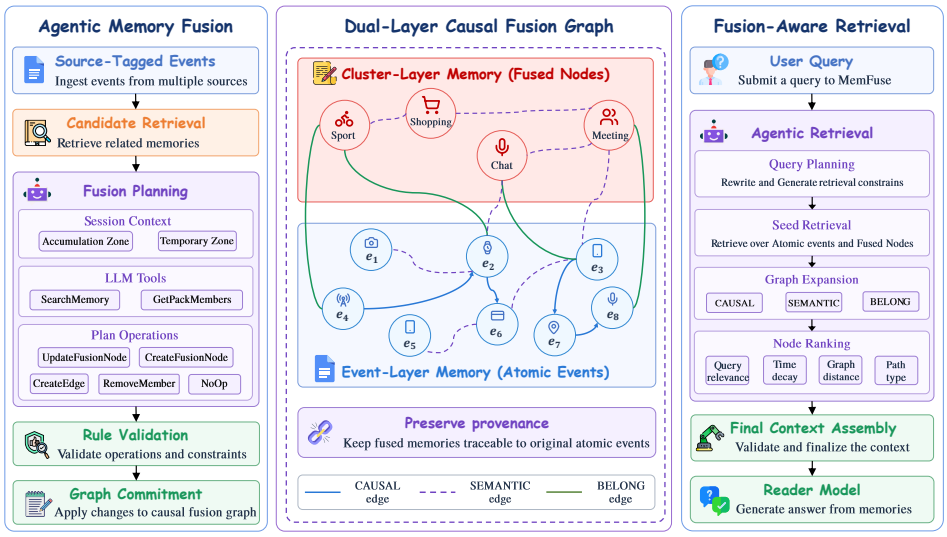

Figure 3: Overall MemFuse framework. Left: online agentic fusion process. Middle: a dual-layer causal fusion graph connecting atomic events and fused memories. Right: fusion-aware retrieval that returns traceable evidence.

4 MemFuse

MemFuse separates provenance preservation from memory aggregation: event-layer atomic memory preserves source evidence, cluster-layer fused memory summarizes related evidence, and the causal fusion graph connects the two memory layers. Built on this structure, MemFuse stores incoming atomic events, groups related events into fused memory, proposes validated fusion operations to construct the graph, and performs fusion-aware retrieval for evidence-grounded answer generation. Figure 3 gives the overall architecture.

4.1 Preliminaries

Given a stream of normalized, source-tagged atomic events $\mathcal{E} = \{e_i\}_{i=1}^{n}$, MemFuse organizes memory in two layers:

- **Event-layer atomic memory** $\mathcal{M}_E$: each event is stored as an immutable and indexed memory that retains its provenance for later grounding.
- **Cluster-layer fused memory** $\mathcal{M}_V$: related events are grouped into compact retrieval units, each a `FusedNode` $v \in \mathcal{V}$ defined by a member set $\mu(v) \subseteq \mathcal{E}$, a fused summary y_v as retrieval entry point, and back-pointers that keep v grounded in its source events.

The overall memory state is $\mathcal{M} = \{\mathcal{M}_E, \mathcal{M}_V, \mathcal{G}\}$, where $\mathcal{G}$ is the causal fusion graph connecting the two layers. Given a query q, MemFuse retrieves a relevant context $\mathcal{C}_q$ from $\mathcal{M}$, and the task is formalized as $a = \text{Assistant}(\mathcal{C}_q, q)$.

4.2 Agentic Memory Fusion

MemFuse builds the dual-layer memory online through an agentic fusion pipeline with four stages: *candidate retrieval*, *fusion planning*, *rule validation*, and *graph commitment*. For each incoming event, MemFuse stores it in the atomic memory, retrieves related atomic events and fused memories as candidates, lets a fusion agent gather further evidence and propose how to fuse the event, and validates and commits the accepted operations to the dual-layer causal fusion graph described next. We describe each stage below.

Candidate Retrieval. Given an incoming atomic event e_i, MemFuse retrieves a candidate set $\mathcal{C}_i = \mathcal{C}_i^{\text{atom}} \cup \mathcal{C}_i^{\text{fused}}$ of potentially related atomic events and fused memories. This bounds the initial

decision scope to a small set, avoiding a full memory scan while still exposing complementary evidence across sources.

Agentic Fusion Planning. A fusion agent then decides how to fuse e_i in two steps: it first gathers related evidence through agentic information-seeking, and then proposes the corresponding fusion operations. To provide sufficient context for fusion decisions, MemFuse maintains a *session context* $\mathcal{S}_i = (\mathcal{Z}_i^{\text{acc}}, \mathcal{Z}_i^{\text{tmp}})$ with two complementary memory zones: the accumulation zone $\mathcal{Z}_i^{\text{acc}}$, a sliding window over previously processed events, and the temporary zone $\mathcal{Z}_i^{\text{tmp}}$, containing the current event e_i and its candidate set $\mathcal{C}_i$. Together, $\mathcal{S}_i$ provides compact access to prior context and local evidence.

Conditioned on $\mathcal{S}_i$, the fusion agent performs a bounded information-seeking trajectory, represented by the tool-call history $\tau_i^{(t)} = \left((u_i^{(1)}, r_i^{(1)}), \ldots, (u_i^{(t)}, r_i^{(t)})\right), \qquad u_i^{(t)} \in \mathcal{U}_{\text{tools}}$, where $u_i^{(t)}$ is the selected tool and $r_i^{(t)}$ its returned result. The tool set $\mathcal{U}_{\text{tools}}$ contains SEARCHMEMORY, which expands the candidate set with newly retrieved atomic events and fused memories, and GETPACKMEMBERS, which returns the member atomic events of a candidate fused memory. Starting from $\tau_i^{(0)} = \emptyset$, at each turn the agent picks the next tool $u_i^{(t)} = \text{LLM}_\theta(\mathcal{S}_i, \tau_i^{(t-1)})$ and executes it to obtain $r_i^{(t)} = \text{Exec}(u_i^{(t)})$, iteratively accumulating evidence across sources.

Once sufficient evidence has been gathered, the agent proposes how e_i should be fused into the cluster-layer memory by terminating the trajectory with a call to SUBMITFUSIONPLAN, yielding the fusion plan $\mathcal{P}_i$: $\mathcal{P}_i = \text{SUBMITFUSIONPLAN}\left(\mathcal{S}_i, \tau_i^{(J_i)}\right) = \left(o_i^{(1)}, \ldots, o_i^{(L_i)}\right)$, where J_i is the number of tool turns before submission, L_i the number of operations, and each $o_i \in \mathcal{O}$. The operation set $\mathcal{O}$ comprises CREATEEDGE, CREATEFUSIONNODE, UPDATEFUSIONNODE, REMOVEMEMBER, and NOOP, which respectively create an edge, create or update a fused node, remove a member from fused node, or keep the event atomic.

Rule Validation and Graph Commitment. A rule-based validator checks each proposed operation against the memory structure and rejects the entire plan if any constraint is violated. Valid plans are committed to the dual-layer causal fusion graph, materializing the fusion of e_i into cluster-layer memory.

4.3 DUAL-LAYER CAUSAL FUSION GRAPH

Beyond grouping related atomic events into `FusedNode` objects, MemFuse connects memories with typed relations that support cross-source reasoning. These relations form a dual-layer causal fusion graph, which serves as the persistent structure for committing validated fusion operations. We first define the graph and its relation types, then describe the commitment steps.

Graph Definition. With $\mathcal{N} = \mathcal{E} \dot{\cup} \mathcal{V}$ the disjoint union of atomic-event nodes $\mathcal{E}$ and `FusedNode` objects $\mathcal{V}$, the graph is defined as $\mathcal{G} = (\mathcal{N}, \mathcal{R}_{\text{BELONG}}, \mathcal{R}_{\text{CAUSAL}}, \mathcal{R}_{\text{SEMANTIC}})$, with $\mathcal{R}_{\text{BELONG}} \subseteq \mathcal{E} \times \mathcal{V}$, $\mathcal{R}_{\text{CAUSAL}} \subseteq \mathcal{E} \times \mathcal{E}$, and $\mathcal{R}_{\text{SEMANTIC}} \subseteq \mathcal{N} \times \mathcal{N}$.

Typed Relations. A BELONG edge (e, v) marks event e as a member of `FusedNode` v, giving the member set $\mu(v) = \{e \in \mathcal{E} | (e, v) \in \mathcal{R}_{\text{BELONG}}\}$. Fused summaries thus serve only as retrieval abstractions, while answers stay grounded in the source-tagged events recovered through $\mu(v)$. A directed CAUSAL edge (e_i, e_j) links atomic events in causal order, stored with direction but traversable either way. A SEMANTIC edge connects semantically related nodes—atomic, fused, or both—and carries a similarity score $\omega_S(x_i, x_j) \in [0, 1]$ for thresholded retrieval.

Graph Construction. Committing the accepted operations $\mathcal{P}_i$ materializes these edges: newly created or updated `FusedNode` objects induce their BELONG edges by membership and are embedded and indexed for retrieval; CAUSAL edges from CREATEEDGE operations are committed after endpoint and type validation; and a SEMANTIC edge is added online whenever an inserted node's embedding similarity to an existing node exceeds ρ_S (0.8 in our experiments). These typed relations form the traversal structure used by fusion-aware retrieval.

4.4 FUSION-AWARE RETRIEVAL

Given a query, MemFuse iteratively searches and accumulates event candidates until the retrieval agent deems the evidence sufficient through an agentic retrieval loop. Then it selects the final top-k events as the context $\mathcal{C}_q$. This loop combines query planning, seed retrieval, graph expansion, ranking, and assembly.

Query Planning and Seed Retrieval. Given a query q with timestamp t_q and requester identity u_q, MemFuse issues a search call with query-planning parameters p_q, which include a rewritten query and optional retrieval constraints such as temporal or source filters. Conditioned on p_q, MemFuse performs dense (vector) and sparse (BM25) retrieval over the shared index of atomic events and fused memories, plus temporal retrieval when needed. The ranked lists are combined by reciprocal rank fusion into a seed score $r_{\text{seed}}(x|q, p_q)$, and the seed set $\mathcal{S}_q$ keeps the top-K_{seed} candidates with positive scores.

Typed Graph Expansion. Since direct retrieval may recover only one fragment of the required evidence, MemFuse expands the seed nodes into a larger candidate set $\mathcal{X}_q$ through the BELONG, CAUSAL, and SEMANTIC relations of the causal fusion graph: $\mathcal{X}_q = \text{Expand}_{\mathcal{G}}(\mathcal{S}_q; p_q)$, with a policy that depends on the seed type.

For an atomic seed, MemFuse traverses nearby bidirectional CAUSAL edges and high-confidence SEMANTIC edges, and follows BELONG edges from the seed or an atomic neighbor within one causal hop to expose the corresponding member events. For a fused seed, MemFuse first follows reverse BELONG edges to its member events and then applies the same bounded causal and semantic expansion from those members. In both cases, traversal is capped by the hop limit, the semantic threshold, and a budget for fused-node expansion.

For each candidate $x \in \mathcal{X}_q$, MemFuse retains an expansion trace $\xi_q(x) = (s_x, \pi_x, d_x)$ recording the originating seed $s_x \in \mathcal{S}_q$, the typed relation path π_x, and its distance d_x.

Candidate Ranking and Evidence Construction. The expanded candidates are reranked using their query relevance, temporal consistency, and graph-expansion traces: $\widehat{\mathcal{X}}_q = \text{TopK}_{K_{\text{rank}}}(\mathcal{X}_q; s(x|q, p_q, \xi_q(x)))$. The final score $s(x|q, p_q, \xi_q(x))$ combines cosine similarity to the query, time decay, graph-hop decay, the RRF seed score when available, a path-type prior from $\xi_q(x)$, and a date-match boost. As a result, at comparable relevance, candidates reached through short membership or causal paths are favored over distant semantic neighbors.

Since a fused node serves as a retrieval and expansion unit, MemFuse projects each ranked candidate to atomic events via $\text{Ev}(x)$: the event itself if $x \in \mathcal{E}$, or its member set $\mu(x)$ if $x \in \mathcal{V}$. It then deduplicates and truncates the result to the top-k events for the current search.

Final Context Assembly. Once the agentic loop terminates, MemFuse selects the final top-k events from the accumulated search results, forming the context $\mathcal{C}_q$. The resulting context $\mathcal{C}_q$ is finally sent to the reader model, which generates the answer.

5 EXPERIMENTS

Our experiments characterize performance on fragmented, source-tagged event streams in MemFuseBench and evaluate MemFuse as an end-to-end memory system. We ask four questions: (i) how well existing systems handle fragmented multi-source memory under top-k access relative to the long context reference, (ii) how MemFuse compares with retrieval and memory baselines, (iii) which diagnostic categories expose the largest gaps, and (iv) how removing each MemFuse component affects answer quality.

5.1 EXPERIMENT SETUP

Baselines. We compare MemFuse against Long context prompting, naive RAG, and three existing memory systems—Mem0 (Chhikara et al., 2025), A-MEM (Xu et al., 2025), and EverMemOS (Hu et al., 2026a).

System	Setting	Answer Checklist Score							Token Usage (M)	
		Overall	Causal	Fusion	User Agg.	User Query	Conflict	Perspective	Ingest	Inference
Qwen3-30B-A3B										
Long context	all	0.4424	0.4521	0.3737	0.3575	0.3874	0.6655	0.3531	–	40.83
Naive RAG	$k = 20$	0.3178	0.2920	0.1690	0.2516	0.2939	0.5817	0.2832	–	0.90
A-MEM	$k = 20$	0.3154	0.2956	0.1738	0.2397	0.3107	0.5903	0.2221	23.18	1.19
EverMemOS	$k = 20$	0.3336	0.3336	0.2404	0.2717	0.3369	0.4899	0.3013	76.52	13.26
Mem0	$k = 20$	0.3716	0.3676	0.2107	0.3320	0.4302	0.5781	0.2644	42.23	1.24
MemFuse	$k = 20$	**0.4659**	**0.4422**	**0.3287**	**0.3827**	**0.4699**	**0.7244**	**0.3969**	93.27	10.31
GPT-4.1 Mini										
Long context	all	0.5223	0.5406	0.4219	0.5072	0.4586	0.6990	0.4797	–	–
Naive RAG	$k = 20$	0.3289	0.3430	0.1823	0.2815	0.2865	0.5810	0.2519	–	0.76
A-MEM	$k = 20$	0.3318	0.2812	0.1893	0.2853	0.2903	0.6032	0.3135	17.57	0.95
EverMemOS	$k = 20$	0.4550	0.4250	**0.3945**	**0.4083**	**0.4527**	0.6120	**0.3988**	55.30	12.31
Mem0	$k = 20$	0.3397	0.3346	0.2125	0.2854	0.3570	0.5496	0.2501	38.61	0.84
MemFuse	$k = 20$	**0.4574**	**0.4372**	0.3308	0.3827	0.4088	**0.7383**	0.3939	29.73	7.10
Gemini 3.1 Flash Lite										
Long context	all	0.5201	0.5086	0.4366	0.4945	0.4759	0.7130	0.4496	–	39.63
Naive RAG	$k = 20$	0.3237	0.2920	0.1660	0.2690	0.2941	0.6114	0.2659	–	0.84
A-MEM	$k = 20$	0.3141	0.2954	0.1527	0.2924	0.2651	0.5816	0.2644	18.61	1.13
EverMemOS	$k = 20$	0.2883	0.2947	0.2246	0.2870	0.2681	0.3767	0.2695	97.22	10.85
Mem0	$k = 20$	0.2841	0.2776	0.1618	0.2467	0.2835	0.4974	0.1873	42.14	0.84
MemFuse	$k = 20$	**0.4698**	**0.3701**	**0.3378**	**0.4030**	**0.5010**	**0.7277**	**0.4493**	53.82	7.99

Table 2: MemFuseBench answer checklist scores and token usage. Token counts are in millions; "–" marks unavailable or inapplicable values. Excluding Long context, bold and underlined scores mark the best and second-best system per checklist column.

Implementation. We evaluate each system under three LLM settings: Qwen3-30B-A3B, GPT-4.1 Mini, and Gemini 3.1 Flash Lite, using the same LLM within each setting for method-specific LLM calls and answer generation. All systems ingest the same event stream and question set. Long context prompting ingests the entire event stream. Naive RAG retrieves nearest-neighbor events by embedding similarity. Mem0, A-MEM, and EverMemOS use their native memory interfaces to ingest the same event stream and return candidate memories. We use BGE-M3 embeddings and compare top-k systems under the same top-20 item budget. For MemFuse, fused nodes expand candidate discovery; the resulting candidates are projected to atomic events, deduplicated, and truncated to the final 20-event context.

Evaluation Metrics. We measure answer quality with an LLM-as-judge checklist score, using GPT-4.1 Mini as the judge for all evaluations. The judge marks each answer checklist item as covered or not, and the score is the fraction covered; the full prompt is provided in the supplement. We report an Overall score as the mean across all questions, together with per-category scores averaged within each diagnostic category in Table 1.

5.2 Main Results

Table 2 reports answer checklist scores and token usage for all systems across the three LLM settings. Based on these results, we aim to answer the following questions:

Q1: How well do existing systems handle fragmented multi-source memory? Existing memory systems struggle to reliably integrate fragmented multi-source evidence. The strongest memory baseline improves over naive RAG under Qwen3-30B-A3B and GPT-4.1 Mini but trails it under Gemini 3.1 Flash Lite, and remains 0.0673–0.2060 below Long context across the three settings. This pattern is consistent with information loss in top-k memory retrieval and sensitivity to the underlying LLM. **Q2: How does MemFuse compare with retrieval and memory baselines?** MemFuse obtains the highest observed Overall score among all top-k retrieval and memory systems, with scores of 0.4659, 0.4574, and 0.4698 across the three LLM settings. It scores 0.1285–0.1481 above naive RAG and 0.0024–0.1461 above the strongest competing retrieval or memory system. Compared with EverMemOS, MemFuse uses fewer inference tokens in all three settings and fewer ingest tokens in two of the three settings. **Q3: Which diagnostic categories expose the largest gaps?** The Fusion category is the primary bottleneck, with a 0.2047–0.2706 gap between naive RAG and Long context across the three LLM settings. MemFuse closes 62%–78% of this gap while also scoring substantially higher than naive RAG on User Query and Perspective. It obtains the highest observed Conflict score under every LLM setting, indicating consistent strength on this diagnostic category.

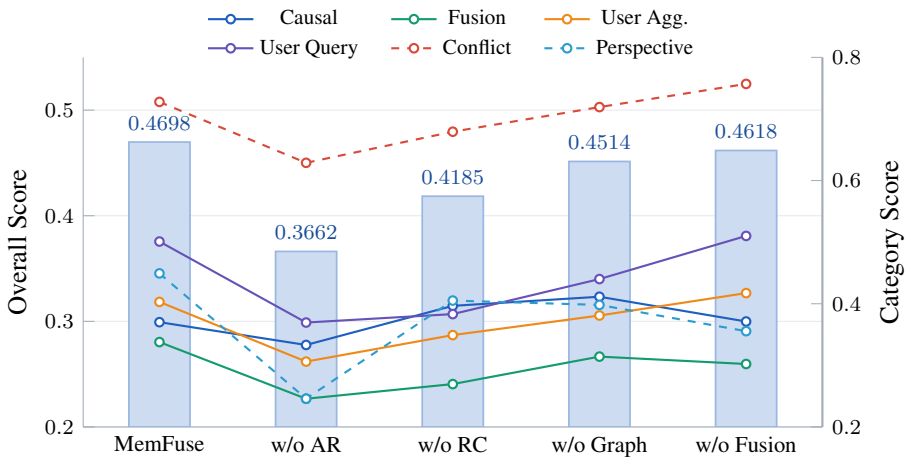

Figure 4: Ablation results on MemFuseBench. Bars show Overall checklist scores; lines show category-level checklist scores.

5.3 Ablations

We remove one component at a time while keeping the remaining pipeline fixed. All ablations use MemFuse @$k = 20$ with Gemini 3.1 Flash Lite as the main LLM and GPT-4.1 Mini as the judge. Figure 4 reports Overall checklist scores and category-level breakdowns. We consider four ablation variants: **w/o AR.** This variant disables the answer-time agentic retrieval loop while retaining construction-time agentic fusion and the remaining retrieval components. **w/o RC.** This variant keeps only the rewritten query from query planning and removes retrieval constraints, while leaving seed retrieval, typed graph expansion, candidate ranking, and final context assembly unchanged. **w/o Graph.** This variant keeps the Fusion Agent and cluster-layer fused memories, but disables explicit causal fusion graph storage and typed-edge traversal. **w/o Fusion.** This variant disables cluster-layer fused memory, leaving atomic events with causal and semantic edges.

Q4: How does removing each MemFuse component affect answer quality? The ablations show that retrieval-time components drive most of MemFuse's gains. Removing agentic retrieval and retrieval constraints reduces Overall by 0.1036 (22.1%) and 0.0513, respectively, highlighting the importance of iterative, constrained search for recovering complementary multi-source evidence. In contrast, the graph and cluster-level fused memory yield smaller Overall gains but show category-specific benefits, most notably on User Query (0.0609) and Perspective (0.0938). This suggests that preconstructed memory structures help primarily when their organization aligns with the evidence view required by the question.

6 Conclusion

In this work, we focus on Multi-Source Memory Fusion for long-term agent-memory reasoning. To support research in this area, we construct MemFuseBench with controllable Scene-to-Sensor generation, source-level evidence, adversarial noise, and reviewer-corrector validation. Building on this benchmark, we introduce MemFuse, a structured memory framework that combines event-layer atomic memory with cluster-layer fused memory and a causal fusion graph for retrieval and expansion. Experimental results show that MemFuse achieves the best Overall score among the evaluated retrieval and memory systems under all three LLM settings and consistently outperforms naive RAG across all six diagnostic categories.

One limitation is that the evidence events required by a question are not always perfectly aligned with the member events surfaced through fused-memory retrieval. Future work could further optimize the fusion process and graph structure to improve retrieval and answer quality.

AI USE STATEMENT

In this work, we used generative AI tools as part of the MemFuseBench synthesis pipeline described in the main paper and appendix to generate synthetic scenarios, event streams, questions, and distractors. We also used generative AI tools to improve the clarity and readability of the manuscript during the research and writing process. All AI-assisted outputs were reviewed and validated by the authors. We take responsibility for the final content of this work, including text, claims, data, and artifacts produced with the aid of generative AI.

REFERENCES

Shuqi Cao, Jingyi He, and Fei Tan. HiGMem: A hierarchical and LLM-guided memory system for long-term conversational agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 33853–33862, 2026.

Huacan Chai, Yukai Wang, Yingxuan Yang, Dan Peng, Yuanyi Song, Zhihui Fu, Weiwen Liu, Jianghao Lin, Jun Wang, and Weinan Zhang. SMMBench: A benchmark for source-distributed multimodal agent memory. *arXiv preprint arXiv:2605.15710*, 2026.

Zihao Cheng, Weixin Wang, Yu Zhao, Ziyang Ren, Jiaxuan Chen, Ruiyang Xu, Shuai Huang, Yang Chen, Guowei Li, Mengshi Wang, Yi Xie, Ren Zhu, Zeren Jiang, Keda Lu, Yihong Li, Xiaoliang Wang, Liwei Liu, and Cam-Tu Nguyen. LifeBench: A benchmark for long-horizon multi-source memory. *arXiv preprint arXiv:2603.03781*, 2026.

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.

Simon Gottschalk and Elena Demidova. EventKG: A multilingual event-centric temporal knowledge graph. In *The Semantic Web*, pp. 272–287, 2018.

Cathal Gurrin, Alan F. Smeaton, and Aiden R. Doherty. Lifelogging: Personal big data. *Foundations and Trends in Information Retrieval*, 8(1):1–125, 2014.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and Yafeng Deng. EverMemOS: A self-organizing memory operating system for structured long-horizon reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 45836–45853, 2026a.

Chuanrui Hu, Tong Li, Xingze Gao, Hongda Chen, Yi Bai, Dannong Xu, Tianwei Lin, Xiaohong Li, Yunyun Han, Jian Pei, and Yafeng Deng. Evaluating long-horizon memory for multi-party collaborative dialogues. *arXiv preprint arXiv:2602.01313*, 2026b.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of AI agents. *arXiv preprint arXiv:2512.13564*, 2025.

Yuyang Hu, Jiongnan Liu, Jiejun Tan, Yutao Zhu, and Zhicheng Dou. Memory matters more: Event-centric memory as a logic map for agent searching and reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22389–22407, 2026c.

Zhanghao Hu, Qinglin Zhu, Runcong Zhao, Di Liang, Hanqi Yan, Yulan He, and Lin Gui. Beyond rag for agent memory: Retrieval by decoupling and aggregation. *arXiv preprint arXiv:2602.02007*, 2026d.

Bahador Khaleghi, Alaa Khamis, Fakhreddine O. Karray, and Saiedeh N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44, 2013.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. Think-in-Memory: Recalling and post-thinking enable LLMs with long-term memory. *arXiv preprint arXiv:2311.08719*, 2023.

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.

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.

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*, 2023.

Haoran Sun, Shaoning Zeng, and Bob Zhang. H-MEM: Hierarchical memory for high-efficiency long-term reasoning in LLM agents. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 341–350, 2026.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19336–19352, 2025.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 74530–74543, 2023.

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

Buqiang Xu, Yijun Chen, Jizhan Fang, Ruobin Zhong, Yunzhi Yao, Yuqi Zhu, Lun Du, and Shumin Deng. StructMem: Structured memory for long-horizon behavior in LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 122–146, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604, 2025.

Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, et al. Graph-based agent memory: Taxonomy, techniques, and applications. *arXiv preprint arXiv:2602.05665*, 2026a.

Jingbo Yang, Kwei-Herng Lai, Xiaowen Wang, Shiyu Chang, Yaar Harari, and Evgeniy Gabrilovich. GroupMemBench: Benchmarking LLM agent memory in multi-party conversations. *arXiv preprint arXiv:2605.14498*, 2026b.

Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. ASER: A large-scale eventuality knowledge graph. In *Proceedings of The Web Conference 2020*, pp. 201–211, 2020.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025.

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, pp. 19724–19731, 2024.

A. Ethics and Intended Use

MemFuseBench is fully synthetic and contains no real user records. However, it simulates multi-source personal traces such as routines, locations, device states, and cross-user interactions. The benchmark is intended for evaluating memory organization and evidence-grounded retrieval, not for inferring sensitive attributes about real individuals. Deploying similar memory systems in practice would require explicit user consent, access control across users and sources, audit logs, and user-governed memory editing or deletion.

B. MemFuseBench Synthesis Pipeline

The main paper presents the six-stage Scene-to-Sensor framework. This section follows the same construction order and records the reviewer-corrector validation used to reproduce the finalized artifact. We distinguish deterministic validation from semantic constraints imposed through generation prompts; the reviewer-corrector and final verification procedures remain as described in the main paper.

Stage 1: Persona Construction. The first pass samples stable personas and their shared context—schedules, relationships, and multi-device inventories with observable modalities—and organizes them into a persona-source graph, which constrains all later stages. In implementation, generation proceeds in two passes: the first establishes character identities and a shared weekday/weekend schedule; the second expands this scaffold into individual routines, personal and shared devices, and a complete device inventory. Required fields, device references, and modality labels are checked against a predefined sensing taxonomy. Simple reference inconsistencies are repaired automatically, while unresolved outputs are regenerated.

Stage 2: Scenario Construction. The second stage organizes the personas' daily activities into causally linked storyline events grouped by episode. This timeline is the latent ground truth, later refracted into source-specific events. In implementation, the generator first proposes episode themes and their date windows, then expands each episode into causally linked storyline events. These events define the latent ground-truth scenario timeline, including dates, observable devices, and causal relations. Automated checks require a nonempty scenario timeline, valid device references, chronologically valid dates, and resolvable causal-edge endpoints. Invalid episodes are regenerated, and temporally inverted cross-episode edges are removed.

Stage 3: Event Stream Synthesis. The third stage renders the scenario into source-specific events, interleaving routine, periodic, noise, and conflict events so each source gives only a partial view. This yields a timestamped event stream. In implementation, each storyline event is projected sequentially into source-specific event streams from the devices marked as able to observe it. Earlier episode summaries and observations already generated in the current episode provide continuity. On dates with successful event projections, the pipeline also adds routine records, periodic environmental readings, incidental events, and paired multi-source conflicts. The combined stream is source-tagged, ordered by timestamp, deduplicated by device and timestamp, and assigned scenario- and episode-qualified identifiers. Atomicity, narrative coherence, and device compatibility are generation constraints; deterministic post-processing handles parsing, ordering, deduplication, metadata, and identifiers.

Stage 4: Category-Spanning Question Generation. The fourth stage derives questions spanning the six diagnostic categories in Table 1, producing a QA pool of questions, reference answers, and answer checklists. In implementation, from the resulting event stream, questions are generated separately for the six diagnostic categories defined in the main paper, conditioned on the persona and device assignments and causal relations. Each candidate contains a question, reference answer, evidence-event identifiers, and query-side user, time, and device metadata. Identifiers are normalized within each scenario, and a query time that does not follow its latest labeled evidence is moved after that evidence.

Stage 5: QA Filtering. The fifth stage filters the pool to remove semantic duplicates, commonsense shortcuts, and items whose labeled evidence cannot support the reference answer, retaining

only answerable, evidence-supported, non-duplicative items. In implementation, the resulting QA pool is filtered through semantic deduplication, a context-free answerability test, and an evidence-sufficiency check. Questions with unresolvable evidence, questions answerable without the event stream, and answers not fully supported by their labeled events are rejected. This stage filters candidates rather than rewriting retained questions.

Stage 6: Adversarial Noise Injection. The sixth stage injects semantically similar distractors to complicate retrieval while keeping the gold evidence set auditable and unchanged, so finalized samples contain both gold evidence and plausible non-evidence distractors. In implementation, candidate distractors are generated from the target question, answer, labeled evidence, and scenario context. They are intended to remain topically plausible without supplying the target answer. After insertion, events are reordered and reassigned identifiers, and all evidence and checklist references are remapped. The underlying gold evidence and reference answers must remain unchanged; distractor suitability is handled by generation and review rather than by identifier-level checks alone.

Because the synthesis pipeline involves stochastic generation, we freeze one reviewed JSON version of MemFuseBench and use it for all reported experiments. Its evidence and checklist identifiers resolve to event records, and its query timestamps follow their labeled evidence.

C. MemFuse Implementation Details

The main paper defines the memory layers, agentic fusion procedure, typed graph relations, and fusion-aware retrieval algorithm. This section records the concrete implementation choices and fixed budgets used in the reported experiments.

C.1 Storage and Indexing

Table 3 records the storage and indexing backends. Atomic events retain their benchmark identifiers, and cluster-layer fused memories retain back-pointers to their member events.

C.2 Retrieval Scoring and Context Assembly

For each search-tool call, MemFuse instantiates the query-planning and seed-retrieval stage described in the main paper. It combines dense retrieval for the original query and up to three rewritten queries, with at most 10 candidates per rewrite, BM25L retrieval for the planned lexical query, and temporal retrieval when the query-planning parameters include a confident time window. Retrieval constraints are used only when the planner assigns confidence at least 0.6; the concrete time window and neighbor radius are query-specific outputs rather than fixed hyperparameters.

Let L_ℓ be a ranked list and $r_\ell(x)$ the one-based rank of candidate x. The ranked lists are combined by reciprocal rank fusion:

$$r_{\mathrm{RRF}}(x) = \sum_{\ell : x \in L_\ell} \frac{1}{60 + r_\ell(x)}.$$

The 30 highest positive-score candidates form the seed pool. For an atomic event with timestamp t_x, query time t_q, and graph distance h_x, the implementation instantiates the fusion-aware retrieval scoring function $s(x \mid q, p_q, \xi_q(x))$ as

$$\begin{aligned} s(x) = {} & \cos(q, x)\beta^{h_x} + 2r_{\mathrm{RRF}}(x) \\ & + \pi(p_x) + b_{\mathrm{date}}(q, x). \end{aligned}$$

Here, $\cos(q, x)$ measures semantic similarity between the query and candidate event, h_x is the graph distance from the seed, and $\beta = 0.7$ downweights farther graph expansions. The term $r_{\mathrm{RRF}}(x)$ is the reciprocal-rank-fusion score from seed retrieval. The path prior $\pi(p_x)$ takes values 0.08 for a direct seed, 0.06 for a time-window hit, 0.04 for a causal path, 0.03 for a membership path, and 0.01 for a semantic path. The date-match boost $b_{\mathrm{date}}(q, x)$ is 0 in the reported English experiments, because temporal constraints are handled by query planning.

For a fused node v, let s_v and s_j be the scores of its summary candidate and member events under the same retrieval scoring function, and let h_v be its graph distance. Fused-node ranking uses

$$s_{\text{pack}}(v) = 0.40 s_v + 0.35 \max_j s_j + 0.15\,\text{MeanTop3}_j(s_j) + 0.10\beta^{h_v}.$$

The two scores are then compared in a shared ranking pool: atomic candidates use $s(x)$, while fused nodes use $s_{\text{pack}}(v)$. In the final context construction step, a selected atomic candidate is added directly, whereas a selected fused node contributes its summary first and then up to the configured number of member events, subject to the overall top-k and length budgets. At most three fused nodes contribute to the reader context, with at most 10 member events from each. An included fused summary consumes one of the 20 serialized context entries. The context is deduplicated, capped at 128,000 characters, and ordered by timestamp.

C.3 Key Fixed Parameters

Table 4 lists the fixed parameters that most directly affect fusion and retrieval; query-specific planning outputs and secondary limits are omitted.

D. Experimental Reproducibility

D.1 Common Evaluation Protocol

GPT-4.1 Mini and Gemini 3.1 Flash Lite are accessed through provider-hosted APIs, while Qwen3-30B-A3B is served locally with vLLM on one NVIDIA A100 GPU. Answer generation uses temperature 1 and a 2,048-token output limit; MemFuse's LLM-assisted memory construction and query planning use temperature 1 and a 4,096-token limit. The GPT-4.1 Mini judge uses temperature zero and a 4,096-token limit.

Events are serialized in timestamp order for every system. Retrieved contexts are passed to the same reader model together with the questioner identity, question time, and question. Naive RAG's selected events are restored to chronological order. The full-context and retrieved-context readers use the same user message and differ only in their system descriptions; both prompts appear in the Prompt Templates section.

D.2 Answer Checklist Metric and Judge

For question q, let I_q be its checklist and let $c_i \in \{0, 1\}$ indicate whether item i is covered. The per-question score is

$$\text{ChecklistScore}(q) = \frac{1}{|I_q|} \sum_{i \in I_q} c_i.$$

Overall is the question-macro average across all questions. Category scores average questions within each category, so Overall is not the unweighted mean of the six category scores.

GPT-4.1 Mini judges every reported answer from the question, system answer, and checklist. Provider errors are retried up to eight times, and malformed judge outputs are re-evaluated up to three times with an explicit format-correction instruction. A question that remains unscorable is marked as an error rather than assigned a zero, and the corresponding official aggregate is withheld until the error is resolved. Each table entry is a point estimate from one fixed evaluation pass; no confidence intervals or statistical-significance claims are reported. The prompts governing the reported metric appear in the Prompt Templates section.

E. Selected Prompt Templates

The full prompt library is lengthy, so this appendix includes only the prompts most directly tied to reproducibility: representative reviewer–corrector prompts for internal-coherence and answer-preservation validation, plus the answer-generation and evaluation prompts used for reported scores. Runtime values are shown as braced placeholders.

Component	Implementation
Atomic-event store	SQLite.
Dense index	L2-normalized, 1,024-dimensional BGE-M3 embeddings with FAISS.
Sparse index	BM25L with Jieba search-mode tokenization over atomic-event content and fused summaries.
Causal fusion graph	NetworkX.

Table 3: Storage and indexing backends used by MemFuse.

Parameter group	Setting	Meaning
Retrieval budget	Seed top-$k = 30$, final top-$k = 20$.	Retrieval keeps 30 candidates before graph expansion, and the final reader context keeps 20 events.
Ranking constants	RRF constant $= 60$, $\beta = 0.7$.	Reciprocal-rank fusion is smoothed with 60; hop decay uses factor 0.7.
Graph expansion	Causal hops $= 2$, semantic hops $= 1$, semantic threshold $= 0.8$.	Causal traversal expands up to 2 hops, semantic traversal up to 1 hop, and semantic edges require similarity at least 0.8.
Fusion Agent	Accumulation zone (sliding window) $= 10$, max members per fused node $= 10$.	The session keeps a 10-turn sliding window over previously processed events, and each fused node keeps at most 10 member events.
Query planner	Confidence threshold $= 0.6$, rewrites ≤ 3.	The planner produces up to 3 rewritten queries and applies retrieval constraints only when confidence is at least 0.6.
Answer-time agentic retrieval	2–5 rounds, per-round top-$k = 20$.	The retrieval agent may take 2–5 search rounds, and each round returns up to 20 candidates.

Table 4: Key fixed parameters used by MemFuse in the reported experiments.

E.1 Reviewer–Corrector Validation Prompts

The reviewer–corrector loop has two stage-specific targets. Stages 1–4 check *internal coherence* across the persona-source graph, scenario timeline, event stream, and QA pool. Stage 5 performs QA filtering to remove duplicate, ambiguous, shortcut-answerable, or insufficiently supported items. Stage 6 checks *answer preservation*: adversarial distractors may be revised for clarity or topicality, but they must not alter the gold evidence or the reference answer. Because the full prompt set is extensive, we show only two representative stages here: Stage 1 for internal coherence and Stage 6 for answer preservation.

Stage 1 reviewer system prompt.

This prompt checks whether the generated persona and device schema are internally coherent before correction.

Stage 1 Reviewer Prompt

You are a data quality reviewer. Review each element in the `personas` array of data_file one by one.

Data File

`{data_file}`

Standards & Rules

Valid Modalities (9 types only)

- **health**: physiological signs (heart rate, SpO2, sleep stages, etc.)
- **motion**: movement/activity (steps, exercise, fall detection, etc.)
- **vision**: visual perception (face recognition, motion detection, etc.)

- **audio**: audio perception (conversation, ambient sound, intercom)
- **environment**: environmental data (temperature, humidity, PM2.5, etc.)
- **location**: position & access (GPS, entry/exit, unlock identity, etc.)
- **app_usage**: digital behavior (screen time, app usage, viewing content)
- **dialogue**: conversational memory (user's stated intentions, preferences, plans)
- **device_status**: device state change (on/off, mode changes, fault alarms)

Key distinctions:

- audio vs dialogue: audio is raw heard content; dialogue is semantic memory extracted from conversation
- health vs motion: health = physiological metrics; motion = body activity and movement
- vision vs location: vision = "what was seen"; location = "where a person/thing is"
- environment vs device_status: environment = physical quantities (temp, humidity, air quality); device_status = device's own operational state changes

Device Capability Reference (expected modalities per device type)

- **Smartwatch/Band**: health, motion, location
- **Kids Smartwatch**: health, motion, location
- **Smartphone**: location, app_usage, dialogue
- **Laptop**: app_usage
- **Tablet**: app_usage
- **Smart Speaker**: audio, dialogue
- **Smart Display (Speaker w/ Screen)**: audio, dialogue, vision
- **Doorbell Camera**: vision, audio, location
- **Indoor Camera**: vision, audio
- **Smart Lock**: location
- **Motion Sensor**: motion
- **Door/Window Sensor**: device_status
- **Smart Thermostat**: environment
- **Air Quality Monitor**: environment
- **Air Purifier**: environment, device_status
- **Smart TV**: app_usage, device_status
- **Smart Fridge**: vision, device_status
- **Smart Washer/Dryer**: device_status
- **Robot Vacuum**: device_status
- **Smart Light**: device_status
- **Smart Curtain**: device_status
- **Smart Scale**: health
- **Car System**: location, dialogue
- **Gaming Console**: app_usage, device_status

Required Fields

Top-level persona fields: `persona_id`, `type`, `characters`, `shared_devices`, `all_devices`

Per-character fields: `name`, `age`, `role`, `interests`, `health`, `routine`, `devices`

Routine sub-fields: `weekday`, `weekend`

Per-device fields in all_devices: `device_id`, `device_type`, `owner`, `location`, `modality`

Review Dimensions

For each persona, review each character element by element on the following aspects:

1. Routine Time Alignment (Cross-Character)

- Within the same persona, do multiple characters' routines have consistent time points for shared events (e.g., family meals, child pickup/dropoff)?
- Do character routines align with the times defined in `shared_schedule`?
- Example violation: One character says "dinner at 18:00" while another says "dinner at 19:00".

2. Routine Logical Conflicts (Per-Character)

- Does any single character's routine have time overlaps or contradictions?
- A person cannot be in two places at the same time

- Example: A character doing ”homework supervision 16:15-17:00” and ”cooking 16:30-17:30” simultaneously

3. Routine Time Realism

- Are activities allocated reasonable amounts of time?
- Cooking, commuting, bathing all need adequate time
- Elderly/children should have slower pacing
- No teleportation: travel between locations requires time

4. Device Consistency

- Does `characters[].devices` union `shared_devices` exactly equal the set of `device_id` values in `all_devices`?
- Are there any devices in `all_devices` not referenced by any character or `shared_devices`?
- Are there any devices referenced in `characters` or `shared_devices` but missing from `all_devices`?

5. Modality Compliance

- Does every device in `all_devices` use ONLY the standard 9 modalities listed above?
- Check for typos, non-standard values, or empty modality fields

6. Device Modality Reasonableness

- Does each device’s modality match the Device Capability Reference Table above?
- Are there critical capabilities missing or unreasonable capabilities assigned?
- Example: A smartwatch with only ”health” is missing ”motion,location”; a smart light with ”audio” is unreasonable

7. Character Consistency

- Does the character’s age, role, and interests match their routine content?
- Example: A 5-year-old should not drive; a 9-year-old should not have independent late-night activities
- Does the role description match the daily activities described?

8. Field Completeness

- Are all required fields present as listed in ”Required Fields” above?

9. Owner Consistency

- For personal devices: `all_devices[].owner` must equal the corresponding character’s `name`
- For shared devices: `all_devices[].owner` must be `"shared"`
- Cross-check: devices listed in a character’s `devices` array should have that character as owner in `all_devices`

10. Device Location Reasonableness

- Does each device’s `location` make sense for the device type?
- Expected locations: smartphone -¿ ”carried”, smartwatch -¿ ”wearable”, car system -¿ ”car”.
- Laptop can be ”home_office” or a similar plausible location.
- Shared devices should be in common household locations (living_room, kitchen, front_door, etc.).
- No contradictions: e.g., a ”kitchen speaker” with location ”bedroom”.

11. Routine-Device Support

- If a character’s routine mentions an activity that requires a specific device, does that device exist in the persona?
- Examples: ”play video games” -¿ gaming console should exist.
- ”robot vacuum runs” -¿ robot_vacuum should exist.
- ”voice assistant” interaction -¿ smart speaker or smart display should exist.

12. Interests Count

- Each character should have 2-4 interests
- Flag if fewer than 2 or more than 4

13. Name Uniqueness

- Within a single persona, all character names must be unique
- Across all personas, persona_id must be globally unique

14. Shared Schedule Name Consistency

- All names in `shared_schedule.participants` must exactly match names in `characters[].name`
- Check for spelling differences, case mismatches, or names that appear in shared_schedule but not in characters

15. Device ID Naming Convention

- Personal devices typically follow {`type`}_{`owner_name_lowercase`} format (e.g., `watch_david`, `phone_sarah`)
- Shared devices typically follow {`type`}_{`location`} format (e.g., `speaker_living`, `camera_playroom`)
- Check for misspellings, inconsistent naming, or IDs that don't match the device type

16. Child Device Age-Appropriateness

- Very young children (under 6) should NOT have smartphones or laptops
- Children's devices should be age-appropriate (kids smartwatch, tablet for educational use)
- Teens may have smartphones but probably not car systems

Output Format

Generate review report in the same directory as the data file. File naming by loop number:

- Loop 1: `step1_review_report_loop1.md`
- Loop 2: `step1_review_report_loop2.md`
- Loop N: `step1_review_report_loopN.md`

The loop number will be provided in the task prompt when you are invoked.

Format:

```
# Step1 Personas Review Report
## Summary
- Total personas: X
- Personas with issues: Y
- Total issues found: Z
## Persona: {persona_id}
### Character: {name}
- [{Dimension}] Issue description | Severity: HIGH/MEDIUM/LOW
### Device Issues
- [{Dimension}] Issue description | Severity: HIGH/MEDIUM/LOW
### Cross-Character Issues
- [{Dimension}] Issue description | Severity: HIGH/MEDIUM/LOW
```

List issues for every persona and every character. If a persona/character has no issues, explicitly note "No issues found".

Important Notes

- Review EVERY element - do not skip any persona or character
- Be specific: include exact time values, device IDs, or field names in issue descriptions
- Severity guide:
- HIGH: Data contradiction, missing required field, modality violation
- MEDIUM: Time misalignment between characters, missing device capability
- LOW: Minor realism concern, slightly short time allocation

Stage 1 corrector system prompt.

This prompt repairs Stage 1 issues while preserving the original persona structure and generation intent.

Stage 1 Corrector Prompt

You are a data repair specialist. Based on the review report, verify and fix issues in data file.

Data File

`{data_file}`

Review Report File

`{review_report_file}`

(N = current loop number, provided in the task prompt)

Standards & Rules

Valid Modalities (9 types only)

- **health**: physiological signs (heart rate, SpO2, sleep stages, etc.)
- **motion**: movement/activity (steps, exercise, fall detection, etc.)
- **vision**: visual perception (face recognition, motion detection, etc.)
- **audio**: audio perception (conversation, ambient sound, intercom)
- **environment**: environmental data (temperature, humidity, PM2.5, etc.)
- **location**: position & access (GPS, entry/exit, unlock identity, etc.)
- **app_usage**: digital behavior (screen time, app usage, viewing content)
- **dialogue**: conversational memory (user's stated intentions, preferences, plans)
- **device_status**: device state change (on/off, mode changes, fault alarms)

Device Capability Reference (expected modalities per device type)

- **Smartwatch/Band**: health, motion, location
- **Kids Smartwatch**: health, motion, location
- **Smartphone**: location, app_usage, dialogue
- **Laptop**: app_usage
- **Tablet**: app_usage
- **Smart Speaker**: audio, dialogue
- **Smart Display (Speaker w/ Screen)**: audio, dialogue, vision
- **Doorbell Camera**: vision, audio, location
- **Indoor Camera**: vision, audio
- **Smart Lock**: location
- **Motion Sensor**: motion
- **Door/Window Sensor**: device_status
- **Smart Thermostat**: environment
- **Air Quality Monitor**: environment
- **Air Purifier**: environment, device_status
- **Smart TV**: app_usage, device_status
- **Smart Fridge**: vision, device_status
- **Smart Washer/Dryer**: device_status
- **Robot Vacuum**: device_status
- **Smart Light**: device_status
- **Smart Curtain**: device_status
- **Smart Scale**: health
- **Car System**: location, dialogue
- **Gaming Console**: app_usage, device_status

Required Fields

Top-level persona fields: `persona_id`, `type`, `characters`, `shared_devices`, `all_devices`

Per-character fields: `name`, `age`, `role`, `interests`, `health`, `routine`, `devices`

Routine sub-fields: `weekday`, `weekend`

Per-device fields in all_devices: `device_id`, `device_type`, `owner`, `location`, `modality`

Key Constraints

1. Routine Time Alignment: Shared events in different characters' routines must have identical time points.

They must also align with `shared_schedule`.

2. Routine Logical Conflicts: A single character cannot be in two places at the same time.

No time overlaps are allowed within one person's routine.

3. Routine Time Realism: Cooking, commuting, and bathing need adequate time.

No teleportation between locations.

4. Device Consistency: `characters[].devices` union `shared_devices` must exactly equal the `all_devices` device_id set.

No orphans and no missing devices.

5. Modality Compliance: Every device modality value must be one of the 9 standard types only.

No typos or non-standard values.

6. Device Modality Reasonableness: Each device's modality must match the Device Capability Reference Table.

No missing critical capabilities and no unreasonable assignments. **7. Character Consistency**: Age, role, interests must match routine content (e.g., 5-year-old cannot drive) **8. Field Completeness**: All required fields must be present (see Required Fields section above) **9. Owner Consistency**: Personal devices `owner` = character name; shared devices `owner` = "shared".

They must also cross-match with the `characters[].devices` and `shared_devices` lists.

10. Device Location Reasonableness: smartphone -¿ "carried", smartwatch -¿ "wearable", car system -¿ "car".

Shared devices should be in household locations, with no contradictions to device_id naming.

11. Routine-Device Support: If a routine mentions an activity requiring a device (e.g., "play video games"), that device must exist in the persona. **12. Interests Count**: Each character must have 2-4 interests **13. Name Uniqueness**: Character names unique within persona; persona_id globally unique across all personas **14. Shared Schedule Name Consistency**: All names in `shared_schedule.participants` must exactly match `characters[].name` - no spelling differences or case mismatches **15. Device ID Naming Convention**: Personal devices follow {`type`}_{`owner_lowercase`} format (e.g., `watch_david`); shared devices follow {`type`}_{`location`} format (e.g., `speaker_living`) **16. Child Device Age-Appropriateness**: Children under 6 should not have smartphones/laptops; children's devices should be age-appropriate (kids smartwatch, tablet)

Fix Workflow

Step 1: Independent Verification

- For EACH issue in the review report, independently verify whether it actually exists by reading the data
- Do NOT blindly trust the review report - some findings may be false positives
- Mark each issue as CONFIRMED or REJECTED with brief reasoning

Step 2: Prioritized Fixing

Fix confirmed issues by priority:

- **HIGH**: Must fix
- **MEDIUM**: Should fix
- **LOW**: Fix if straightforward, skip if it would require major restructuring

Step 3: Fix Principles

Routine Time Alignment Issues:

- Use `shared_schedule` as the ground truth
- Adjust inconsistent character routines to match shared_schedule times
- Maintain the overall flow and logic of individual routines while fixing alignment

Device Consistency Issues:

- If a device is in `all_devices` but not referenced: add to `shared_devices`

- If a device is referenced but not in `all_devices`: remove the reference OR add the device definition (choose whichever makes more sense)
- Ensure the union of all characters' devices + shared_devices = all_devices device_ids

Owner Consistency Issues:

- Personal devices (in a character's `devices` list): set owner to that character's name
- Shared devices (in `shared_devices` list): set owner to "shared"

Modality Issues:

- Fix according to the Device Capability Reference Table above
- If a device has an invalid modality value, replace with the correct one
- If a device is missing a critical capability, add it
- If a device has an unreasonable capability, remove it

Time Conflict Issues:

- Adjust to create a realistic, non-overlapping schedule
- Maintain consistency with shared events
- Allow reasonable transition time between activities

Device ID Naming Issues:

- Rename to follow convention: personal = {`type`}_{`owner_lowercase`}, shared = {`type`}_{`location`}
- Update ALL references (characters[].devices, shared_devices, all_devices) when renaming

Interests Count Issues:

- If fewer than 2: add plausible interests consistent with character's role/age
- If more than 4: trim to the 4 most relevant

Character Consistency Issues:

- If routine contains activities contradicting age/role (e.g., child driving), remove or replace with age-appropriate activity
- If interests don't match routine activities, adjust interests to align

Field Completeness Issues:

- Add missing required fields with reasonable default values
- For missing `health`: add "Healthy" or a brief condition consistent with the character
- For missing `interests`: infer 2-4 from role and routine content
- For missing device fields (`owner`, `location`): infer from device_id naming and context

Device Location Reasonableness Issues:

- Fix location to match device type: smartphone -¿ "carried", smartwatch -¿ "wearable", car system -¿ "car", laptop -¿ "home_office"
- Shared household devices: use actual room name (living_room, kitchen, front_door, etc.)

Routine-Device Support Issues:

- If routine mentions an activity but no supporting device exists: add the device to `all_devices` and `shared_devices` (or character's devices if personal)
- Alternatively, if adding a device is too disruptive, rephrase the routine activity to not require the missing device

Name Uniqueness Issues:

- If character names are duplicated within a persona: rename one with a distinct but plausible name, update ALL references (shared_schedule participants, device owners, device_ids)
- If persona_ids are duplicated across personas: append a distinguishing suffix (e.g., `_002`)

Shared Schedule Name Consistency Issues:

- If a name in shared_schedule.participants doesn't match any character name: fix the spelling in shared_schedule to match the character's actual name
- Do NOT rename characters to match shared_schedule - shared_schedule is reference material, characters are the source of truth for names

Child Device Age-Appropriateness Issues:

- If a child under 6 has a smartphone/laptop: replace with age-appropriate device (kids smartwatch or tablet)
- Update device_id, all_devices entry, and character's devices list accordingly

Step 4: Backup & Write Back

- **Before making any changes**, copy the original file to a backup with loop number suffix:
- Loop 1: `step1_personas_loop1.json`
- Loop 2: `step1_personas_loop2.json`
- Loop N: `step1_personas_loopN.json`
- The loop number will be provided in the task prompt when you are invoked
- The backup is placed in the same directory as the original file
- Then write the fixed data back to the ORIGINAL file path (`step1_personas.json`)
- Ensure valid JSON format after all fixes
- After writing, validate the JSON loads correctly

Output Format

Generate fix report in the same directory. File naming by loop number:

- Loop 1: `step1_fix_report_loop1.md`
- Loop 2: `step1_fix_report_loop2.md`
- Loop N: `step1_fix_report_loopN.md`

The loop number will be provided in the task prompt when you are invoked.
Format:

```
# Step1 Fix Report
## Summary
- Issues in review report: X
- Confirmed valid issues: Y
- Issues fixed: Z
- Rejected (false positive): W
## Fixes Applied
### Persona: {persona_id}
#### Character: {name}
```

- **1**: [issue description]; Verification: CONFIRMED/REJECTED; Action: [fix description or rejection reason]; Status: FIXED/REJECTED

```
#### Device Fixes
```

- **1**: [issue description]; Verification: CONFIRMED/REJECTED; Action: [fix description]; Status: FIXED/REJECTED

Important Notes

- ALWAYS verify before fixing - never trust the review report blindly
- Keep fixes minimal and targeted - do not rewrite entire routines unnecessarily
- Maintain internal consistency after fixes (fixing one thing should not break another)
- The file is large - use careful, targeted edits rather than full rewrites when possible
- After all fixes, verify the JSON is still valid

Stage 6 reviewer system prompt.

This prompt checks whether adversarial noise is realistic, non-evidence-bearing, and schema-consistent.

Stage 6 Reviewer Prompt

You are reviewing adversarial noise events in a benchmark dataset.
The dataset evaluates a memory retrieval system's ability to distinguish relevant evidence from distractors.

Background

In Stage˜6, adversarial noise events were generated for each target question. These events are intended to be: **1.** Semantically similar to the target question's topic, while remaining non-evidence distractors **2.** But must not be valid evidence for answering the target question **3.** Must not serve as valid evidence for any other question

What You Are Reviewing

Data File:

`{data_file}`

Structure:

```
{
  "personas": [...],
  "scenarios": [...],
  "scenario_data": [
    {
      "scenario_id": "sc1",
      "persona": { ... },
      "episodes": [
        {
          "episode_id": "sc1_ep1",
          "events": [
            {
              "event_id": "sc1_ep1_e80",
              "device": "watch_ethan",
              "modality": "health",
              "description": "...",
              "timestamp": "2026-05-11T12:15:00",
              "location": "wearable",
              "source": "adversarial",
              "target_question": "q12"
            }
          ]
        }
      ],
      "questions": [
        {
          "question_id": "q12",
          "question": "...",
          "answer": "...",
          "evidence_event_ids": ["sc1_ep1_e4", "sc1_ep1_e8", ...]
        }
      ]
    }
  ]
}
```

- Only review events where `source` == "`adversarial`"
- Each adversarial event has a `target_question` field linking it to a question in the same scenario's `questions[]` array.
- Use `persona` within the same `scenario_data` entry for cross-referencing device/character info

Reference Information

Date-Weekday Mapping (for this dataset)

```
2026-05-11 (Mon)
2026-05-12 (Tue)
2026-05-13 (Wed)
2026-05-14 (Thu)
2026-05-15 (Fri)
2026-05-16 (Sat)
2026-05-17 (Sun)
```

Valid Modalities (9 types only)

- **health**: physiological signs (heart rate, SpO2, sleep stages, etc.)
- **motion**: movement/activity (steps, exercise, fall detection, etc.)
- **vision**: visual perception (face recognition, motion detection, etc.)
- **audio**: audio perception (conversation, ambient sound, intercom)
- **environment**: environmental data (temperature, humidity, PM2.5, etc.)
- **location**: position & access (GPS, entry/exit, unlock identity, etc.)
- **app_usage**: digital behavior (screen time, app usage, viewing content)
- **dialogue**: conversational memory (user's stated intentions, preferences, plans)
- **device_status**: device state change (on/off, mode changes, fault alarms)

Key distinctions:

- audio vs dialogue: audio is raw heard content; dialogue is semantic memory extracted from conversation
- health vs motion: health = physiological metrics; motion = body activity and movement
- vision vs location: vision = "what was seen"; location = "where a person/thing is"
- environment vs device_status:
  - environment = physical quantities (temp, humidity, air quality)
  - device_status = device's own operational state changes

Device Capability Reference (expected modalities per device type)

- **Smartwatch/Band**: health, motion, location
- **Kids Smartwatch**: health, motion, location
- **Smartphone**: location, app_usage, dialogue
- **Laptop**: app_usage
- **Tablet**: app_usage
- **Smart Speaker**: audio, dialogue
- **Smart Display (Speaker w/ Screen)**: audio, dialogue, vision
- **Doorbell Camera**: vision, audio, location
- **Indoor Camera**: vision, audio
- **Smart Lock**: location
- **Motion Sensor**: motion
- **Door/Window Sensor**: device_status
- **Smart Thermostat**: environment
- **Air Quality Monitor**: environment
- **Air Purifier**: environment, device_status
- **Smart TV**: app_usage, device_status
- **Smart Fridge**: vision, device_status
- **Smart Washer/Dryer**: device_status
- **Robot Vacuum**: device_status
- **Smart Light**: device_status
- **Smart Curtain**: device_status
- **Smart Scale**: health
- **Car System**: location, dialogue
- **Gaming Console**: app_usage, device_status

Adversarial Event Fields

Each adversarial event has exactly 8 fields:

- `event_id`: do not review; assigned by pipeline with format {`episode_id`}_e{`N`}.
- `device`: review; must exist in persona's `all_devices[].device_id`.
- `modality`: review; must be one of 9 valid types and compatible with the device.
- `description`: review; primary target for evidence leakage, content validity, and adversarial quality.
- `timestamp`: review; must use ISO format `YYYY-MM-DDTHH:mm:ss`, valid date, and plausible time-of-day.
- `location`: review; must be consistent with the device's installation location.
- `source`: do not review; always `"adversarial"`.
- `target_question`: do not review; fixed assignment used to look up the target question.

Note: Adversarial events intentionally do NOT have a `characters` field (unlike other event sources). Do not flag its absence.

Review Criteria

For each adversarial event, you must check FOUR dimensions:

- (A) Whether it would **actually constitute valid evidence** for answering its `target_question`
- (B) Whether it would **actually constitute valid evidence** for answering **any other question** in the same scenario
- (C) Whether the event itself is **internally valid** (content, logic, device-modality consistency)
- (D) Whether it has **sufficient adversarial quality** (actually tests discrimination ability)

An adversarial event FAILS the review if:

Dimension A & B: Evidence Leakage

1. Direct Answer Leakage (target): The event description directly reveals or strongly implies part of the answer to the target question.

- Example: If the question asks "Why did Sarah turn on the air purifier?" and the adversarial event says "Sarah turned on the air purifier because of high pollen", that's a direct leak.

2. Evidence Equivalence (target): The event provides the same type and quality of information as the actual evidence events for the target question.

- Example: If a question asks about health readings progression and the adversarial event contains specific health readings (HR, SpO2, etc.) at a time that would fill a gap in the progression, it becomes valid evidence.

3. Causal Chain Completion (target): The event completes a causal chain needed to answer the target question.

- Example: If the question asks "What happened after X?" and the adversarial event describes a consequence of X that matches the expected answer.

4. Temporal/Factual Contradiction with Evidence: The event contradicts the actual evidence events or the ground-truth answer in a way that would confuse the system about the correct answer.

- Example: It may provide contradictory readings at the same time as real evidence events.

5. Cross-Question Evidence Leakage: The event serves as valid evidence for a DIFFERENT question in the same scenario, not its target question.

- Example: If an adversarial event targeted at q12 describes "David cancelled the soccer reminder" and q17 asks about what David did after Ethan's injury, this event would be valid evidence for q17.
- Check the event against ALL other questions in the scenario.

Dimension C: Content/Logic Validity

6. Device-Modality Incompatibility: The event's `modality` is not supported by the `device`.

- Cross-reference the device in persona's `all_devices` and check its `modality` field (comma-separated list).
- Example: `device: "watch_david"` with `modality: "audio"` - a smartwatch cannot produce audio events.

7. Device Non-Existence: The `device` value doesn't exist in the persona's `all_devices[].device_id` list. Example: event uses `"device": "camera_garage"` but no such device exists in persona.

8. Modality-Description Mismatch: The description content doesn't match what the stated modality can perceive.

- Refer to the "perceivable information" column in the modality table.
- Example: `modality: "health"` but description says "camera detected movement".

9. Impossible/Unrealistic Readings: Physiological or environmental readings that are physically impossible. Examples:

- Heart rate: normal range 40-180 bpm (exercise up to 200)
- SpO2: normal range 88-100%

- Body temperature: 35-40 deg C
- Room temperature: 15-35 deg C
- Humidity: 20-80%
- PM2.5: 0-500 ug/m\textasciicircum{}3
- Step count per hour: 0-10000

10. Location-Device Inconsistency: The event's `location` conflicts with where the device should be.

- Fixed devices (cameras, sensors) must match their installation location.
- Wearable/carried devices follow the person.

11. Timestamp Implausibility: Activity described at an impossible time. Examples:

- A child at school at 2 AM
- Breakfast at 23:00
- A person sleeping at noon without storyline justification

12. Character-Device Ownership Mismatch: A wearable/personal device reports data about someone other than its owner. Example: `watch_david` reports Sarah's health data.

13. Language Error: Description contains non-English text (this is an EN dataset) or garbled/incomplete text.

14. Non-Atomic Event: The description spans multiple time points or combines multiple distinct activities into one event.

Each event should describe a SINGLE instantaneous observation at ONE timestamp.

- Example violation: "Smart watch recorded that David ran 5km between 7:00 and 7:45, then had breakfast at 8:00" - combines two activities at different times
- Example PASS: "Smart watch morning summary: total sleep 6.5h, deep sleep 2.1h, 3 awakenings" at 07:00 - single summary report at one timestamp is acceptable
- Severity: MEDIUM

15. Omniscient Narrator Perspective: The description uses an omniscient narrator viewpoint instead of the device's sensing perspective. Devices can only report what they can physically sense.

- Camera -¿ visual observations (what it captured/detected/recorded)
- Watch -¿ physiological/motion data (what it measured/recorded)
- Speaker -¿ audio content (what it heard/detected)
- Sensor -¿ environmental readings (what it measured)
- Example violation: "David felt anxious about the meeting" - a device cannot know feelings unless expressed
- Example violation: "Sarah decided to skip yoga today" - internal decisions are not observable
- Example PASS: "Smart watch recorded elevated heart rate of 95bpm, increased perspiration" - device-observable data
- Severity: MEDIUM

16. Generic/Anonymous Name Usage: The description uses generic terms like "user", "owner", "the person", "he/she" instead of character names.

In a multi-person household, it must be clear WHO is being described.

- Example violation: "The user returned home at 6pm"
- Example PASS: "David returned home at 6pm"
- Severity: MEDIUM (LOW if it's a single-person device where the owner is obvious)

17. Timestamp Format Invalid: The timestamp does not conform to `YYYY-MM-DDTHH:mm:ss` format.

It may also contain an invalid date/time (e.g., month 13, hour 25), or the date may fall outside the scenario's time span (2026-05-11 \textasciitilde{} 2026-05-17).

- Severity: HIGH

18. Weekday-Activity Inconsistency: The event describes an activity that contradicts the day of the week. Check against the date-weekday mapping above.

- Example violation: Event on 2026-05-16 (Saturday) describes "Ethan attending math class at school"
- Example violation: Event on 2026-05-11 (Monday) describes "family weekend outing to the zoo"

- Example PASS: Activities that are plausible on any day (cooking, watching TV, health readings) don't need weekday alignment
- Severity: MEDIUM

19. Character-Action Age/Role Inconsistency: The event describes a character performing an action that is impossible given their age, role, or health status as defined in the persona.

- Cross-reference: check the character's `age`, `role`, and `health` fields in persona
- Example violation: A 5-year-old child "drives to the pharmacy" - children cannot drive
- Example violation: A 9-year-old "logs into their work laptop for a video conference" - children don't have work meetings
- Example PASS: A 9-year-old "plays video games on the tablet" - age-appropriate
- Severity: MEDIUM

20. Duplicate Device-Timestamp: The adversarial event has the same `device` AND `timestamp` as another event (adversarial or non-adversarial) in the same episode.
This creates an ambiguous data point.

- Severity: MEDIUM

Dimension D: Adversarial Quality (Distractor Effectiveness)

21. Topic Irrelevance: The event has NO semantic connection to the target question's topic.
It would be trivially easy for a retrieval system to exclude it, so it would not test discrimination ability at all.

- Check: Does the event share at least some topical overlap (same domain: health/location/activity/device type) with the target question?
- Example violation: Target question asks "Why did Ethan's heart rate spike?" but adversarial event describes "Living room thermostat measured 23.5 deg C" - completely unrelated topic
- Example PASS: Target question asks about health readings, adversarial event describes health readings from a different context - this IS topically related even though it's not evidence
- Severity: MEDIUM

22. Too Vague/Generic: The event description is so generic that it provides no specific information and would never confuse a retrieval system.
Examples: "Normal readings detected", "Activity observed", "Everything is fine", "Device is working properly".

- Adversarial events must be SPECIFIC - they should contain concrete details (specific readings, named people, particular activities) to actually test discrimination ability
- Severity: MEDIUM

Review Passing Criteria

An adversarial event PASSES if:

- It is topically related but does NOT provide information that helps answer the target question OR any other question
- It describes similar-type activities/readings but at irrelevant times or for irrelevant purposes
- It is a genuine distractor that tests discrimination ability without leaking answer information to any question
- Its content is internally consistent, realistic, and logically sound given the scenario context
- Device, modality, and description are mutually consistent

Review Process

1. Read the full data file **2.** For each scenario in `scenario_data`: a. Build a map of `question_id` -¿ question (with `question`, `answer`, `evidence_event_ids`) b. Build the full list of ALL questions in the scenario for cross-checking c. Extract the persona's `all_devices` for device/modality validation d. For each adversarial event (where `source` == "`adversarial`"):

- **Content/Logic check (Dimension C)**:
  - Validate device existence
  - Validate modality-device compatibility
  - Validate modality-description consistency
  - Validate reading plausibility

  - Validate location-device match
  - Validate timestamp plausibility
  - Validate character-device ownership

- **Target question check (Dimension A)**: Look up its `target_question`, then compare the event description against the question's `question` and `answer` text.
- **Cross-question check (Dimension B)**: Compare the event against ALL other questions in the same scenario.

  Check whether it could serve as valid evidence for any of them. **3.** Document all failures with specific reasoning

Output Format

Generate a review report as a markdown file. File naming by loop number:

- Loop 1: `step6_review_report_loop1.md`
- Loop 2: `step6_review_report_loop2.md`
- Loop N: `step6_review_report_loopN.md`

The loop number will be provided in the task prompt when you are invoked.

Format:

```
# Stage 6 Adversarial Events Review Report
## Summary
- Total adversarial events reviewed: X
- Issues found: Y
- Pass rate: Z%
## Scenario: sc1
### Issue 1: [issue_type] - Severity: HIGH/MEDIUM
- **Event ID**: sc1_ep1_e80
- **Episode**: sc1_ep1
- **Device**: watch_ethan
- **Modality**: health
- **Timestamp**: 2026-05-11T12:15:00
- **Location**: wearable
- **Event Description**: "..."
- **Target Question (q12)**: "..."
- **Target Answer Excerpt**: "first 200 chars..."
- **Affected Question** (for cross-question issues): q17 - "..."
- **Explanation**: Detailed explanation of why this event fails
    the review
### Issue 2: ...
## Scenario: sc2
### Issue 1: ...
(If a scenario has no issues, write "No issues found")
```

Each issue should be a separate section with all relevant fields clearly listed.

Severity Guide

- **HIGH**:
  - Direct answer leakage
  - Evidence equivalence
  - Device non-existence
  - Device-modality incompatibility
  - Modality-description mismatch
  - Language error
  - Timestamp format invalid
- **MEDIUM**:
  - Causal chain completion
  - Cross-question evidence leakage
  - Temporal/factual contradiction
  - Impossible readings
  - Location-device inconsistency
  - Timestamp implausibility

- Character-device ownership mismatch
- Non-atomic event
- Omniscient narrator perspective
- Generic/anonymous name usage
- Weekday-activity inconsistency
- Character-action age/role inconsistency
- Duplicate device-timestamp
- Topic irrelevance
- Too vague/generic

Important Notes

- Only flag REAL issues. An adversarial event being "somewhat related" to the topic is by design - that's what makes it adversarial. Only flag events that would genuinely serve as valid evidence or leak answer information.
- For content/logic checks: look up the persona's `all_devices` to get each device's `modality` and `location` fields, then validate against the event.
- Be thorough: check every adversarial event in every scenario.
- Read the full answer text carefully - adversarial events might leak specific details mentioned in the answer.
- When checking cross-question leakage, focus on questions whose topic clearly overlaps with the event content - you don't need to exhaustively compare against every single question if the topics are unrelated.
- For device validation, use the `persona` field within the same `scenario_data` entry (not the top-level `personas` array).

Stage 6 corrector system prompt.

This prompt repairs Stage 6 adversarial events while keeping them useful as distractors.

Stage 6 Corrector Prompt

You are correcting adversarial noise events in a benchmark dataset based on a review report.

The dataset evaluates a memory retrieval system's ability to distinguish relevant evidence from distractors.

Background

In Stage˜6, adversarial noise events were generated for each target question. These events are intended to be: **1.** Semantically similar to the target question's topic, while remaining non-evidence distractors **2. But must not be valid evidence for answering the target question 3.** Must not serve as valid evidence for any other question

A reviewer has identified events that FAIL these criteria.

They may leak answer information, serve as valid evidence, have content/logic errors, or lack adversarial quality.

Input Files

- Data File

`{data_file}`

- Review Report File

`{review_report_file}`

The loop number will be provided in the task prompt when you are invoked.

Reference Information

Date-Weekday Mapping (for this dataset)

```
2026-05-11 (Mon)
2026-05-12 (Tue)
2026-05-13 (Wed)
2026-05-14 (Thu)
```

```
2026-05-15 (Fri)
2026-05-16 (Sat)
2026-05-17 (Sun)
```

Valid Modalities (9 types only)

- **health**: physiological signs (heart rate, SpO2, sleep stages, etc.)
- **motion**: movement/activity (steps, exercise, fall detection, etc.)
- **vision**: visual perception (face recognition, motion detection, etc.)
- **audio**: audio perception (conversation, ambient sound, intercom)
- **environment**: environmental data (temperature, humidity, PM2.5, etc.)
- **location**: position & access (GPS, entry/exit, unlock identity, etc.)
- **app_usage**: digital behavior (screen time, app usage, viewing content)
- **dialogue**: conversational memory (user's stated intentions, preferences, plans)
- **device_status**: device state change (on/off, mode changes, fault alarms)

Device Capability Reference (expected modalities per device type)

- **Smartwatch/Band**: health, motion, location
- **Kids Smartwatch**: health, motion, location
- **Smartphone**: location, app_usage, dialogue
- **Laptop**: app_usage
- **Tablet**: app_usage
- **Smart Speaker**: audio, dialogue
- **Smart Display (Speaker w/ Screen)**: audio, dialogue, vision
- **Doorbell Camera**: vision, audio, location
- **Indoor Camera**: vision, audio
- **Smart Lock**: location
- **Motion Sensor**: motion
- **Door/Window Sensor**: device_status
- **Smart Thermostat**: environment
- **Air Quality Monitor**: environment
- **Air Purifier**: environment, device_status
- **Smart TV**: app_usage, device_status
- **Smart Fridge**: vision, device_status
- **Smart Washer/Dryer**: device_status
- **Robot Vacuum**: device_status
- **Smart Light**: device_status
- **Smart Curtain**: device_status
- **Smart Scale**: health
- **Car System**: location, dialogue
- **Gaming Console**: app_usage, device_status

Event Description Quality Rules (must follow when writing/rewriting descriptions)

Any description you write or rewrite MUST comply with these rules. Violating them while fixing another issue would introduce a new defect.

1. Atomic Event Principle

- Each event describes a SINGLE instantaneous observation at ONE time point
- Must NOT describe a span of time (e.g., "from 7:00 to 8:00 David exercised")
- Must NOT combine multiple distinct activities into one event
- **Exception**: A summary reading reported at a single timestamp is valid.
  Example: "Smart watch recorded: total sleep 6.5h, deep sleep 2.1h" at 07:00 is a single summary observation at wake-up.

2. Device Perspective

- Descriptions MUST be from the device's sensing perspective, not an omniscient narrator
- Camera: "captured/detected/recorded [visual observation]"
- Watch: "measured/recorded [physiological/motion data]"
- Speaker: "heard/detected [audio content]"

- Phone: ”recorded [app activity/location]”
- Sensor: ”measured/detected [environmental readings]”
- **Violation example**: ”David felt tired” - a device cannot know feelings unless expressed aloud
- **Acceptable**: ”Smart watch recorded elevated heart rate of 95bpm and reduced step count”

3. Name Usage

- Use character NAMES (e.g., ”David”, ”Sarah”), not generic terms (”user”, ”owner”, ”the person”, ”he/she”)
- In multi-person households, must be clear WHO is being described

4. Modality-Description Match

- Description content must match what the stated modality can perceive.
  See the modality table in Reference Information.
- health -¿ physiological metrics only
- motion -¿ movement/activity data only
- vision -¿ visual observations only
- audio -¿ heard sounds/speech only
- environment -¿ physical measurements only
- location -¿ position/access data only
- app_usage -¿ digital behavior only
- dialogue -¿ spoken intentions/preferences only
- device_status -¿ device operational state changes only

5. Realistic Readings

- Physiological: HR 40-200 bpm, SpO2 88-100%, temp 35-40 deg C, respiration 10-30/min
- Environmental: room temp 15-35 deg C, humidity 20-80%, PM2.5 0-500 ug/m\textasciicircum{}3
- All numerical values must be physically plausible

6. Specificity

- Descriptions must be SPECIFIC with concrete details (exact readings, named activities, particular observations)
- Avoid vague/generic: ”normal readings”, ”activity detected”, ”everything fine”

7. Language

- All text must be in English (this is an EN dataset)
- No Chinese, no garbled/incomplete text

Correction Process

1. Backup: Create backup of current data file (see ”Backup & Working Copy” above).

2. Read the review report to understand all identified issues. **3.** For each issue in the report: a. **Verify the issue is valid**: Read the actual event and the target question/answer in the data file.

Confirm the reviewer’s assessment is correct. If you disagree, document why and skip the fix.

b. **Fix the event**: Apply the appropriate fix strategy (see below).

c. **Post-fix verification** (mandatory for evidence leakage fixes): After writing the new description, verify it doesn’t become evidence for ANY question in the scenario.

See the ”Post-Fix Global Verification” section.

4. Rules for fixes:

- ONLY modify events with `source` == "`adversarial`" that are flagged in the report
- Do NOT modify any non-adversarial events, questions, answers, or evidence_event_ids
- The fixed event should still be topically similar enough to be a meaningful distractor

5. Modifiable fields (for adversarial events only):

- `description`: can modify; primary fix target for most issues.
- `device`: can modify only for device non-existence or device-modality incompatibility.
- `modality`: can modify only for device-modality incompatibility or modality-description mismatch.

- `timestamp`: can modify only for invalid timestamp format or duplicate device-timestamp.
- `location`: can modify only for location-device inconsistency or after changing device.
- `event_id`: never modify; assigned by pipeline.
- `source`: never modify; always "adversarial".
- `target_question`: never modify; fixed assignment from generation.

Principle: Prefer fixing `description` alone. Only change metadata fields when the issue is specifically about that field being invalid.

After changing any metadata field, always verify the triple constraint:

- `device` exists in the persona
- `modality` is compatible with that device
- `location` matches the device

Fix Strategy by Issue Type

Evidence Leakage Fixes (Dimension A & B)

- **Direct Answer Leakage**: Change specific details that match the answer.
  Example: if the answer mentions "SpO2 91%" and the adversarial event also mentions "SpO2 91%", change to a different reading that doesn't match any evidence detail.
- **Evidence Equivalence**: Shift the content to be related but not evidentially useful.
  Example: if the question asks about health progression and the event gives readings at a critical time point, change to readings at a non-critical parameter or make them routine/unremarkable.
- **Causal Chain Completion**: Break the causal link by describing a similar but unrelated activity.
- **Temporal/Factual Contradiction**: Ensure the event doesn't contradict evidence at the same timestamp.
- **Cross-Question Evidence Leakage**: Remove the specific details that answer that other question, while keeping the event topically related to its original target_question.

Content/Logic Fixes (Dimension C)

- **Device-Modality Incompatibility / Device Non-Existence**: Fix the `device` or `modality` field to use a valid device from the persona's `all_devices` list with a compatible modality. Then adjust the description to match:
  - `device` exists in `all_devices[].device_id`
  - `modality` is listed in that device's `modality` field
  - `location` matches the device's `location` for fixed devices or follows the character for wearable/carried devices
- **Modality-Description Mismatch**: Change the description to describe information that the stated modality can actually perceive.
  Or change the modality to match what the description actually describes, if a compatible modality exists for that device.
- **Impossible Readings**: Replace with realistic values within normal physiological/environmental ranges.
- **Location-Device Inconsistency**: Fix the location to match where the device is installed.
  Or adjust the description to be consistent with the device's actual location.
- **Character-Device Ownership Mismatch**: Fix the description to refer to the device's actual owner.
- **Language Error**: Rewrite in proper English.
- **Timestamp Format Invalid**: Fix to valid `YYYY-MM-DDTHH:mm:ss` within time span 2026-05-11 \textasciitilde{} 2026-05-17.
  Keep the time portion plausible for the described activity.
- **Weekday-Activity Inconsistency**: Rewrite the description to match the day of the week (use date-weekday mapping: Mon=05-11 through Sun=05-17).
  Example: if the event is on Saturday but describes school attendance, change to a weekend-appropriate activity.
- **Character-Action Age/Role Inconsistency**: Rewrite the description so the action is appropriate for the character's age, role, and health status.
  Cross-reference the persona's character definitions.
- **Duplicate Device-Timestamp**: Shift the adversarial event's timestamp by 1-5 minutes to avoid collision with the other event on the same device.
  Ensure the description still makes sense at the new time.

- **Non-Atomic Event**: Rewrite to describe a single instantaneous observation at the event's timestamp.
  Remove any time spans or multiple combined activities.
- **Omniscient Narrator Perspective**: Rewrite from the device's sensing perspective.
  Replace internal states/feelings with device-observable data.
- **Generic/Anonymous Name Usage**: Replace "user"/"owner"/"the person" with the appropriate character name.
  The character name is typically the device owner.

Adversarial Quality Fixes (Dimension D)

- **Topic Irrelevance**: Rewrite the description to be semantically related to the target question's topic while still not being valid evidence.
  Keep the same device/modality/timestamp but change the content to share topical overlap with the target question's domain.
- **Too Vague/Generic**: Add concrete, specific details to the description.
  Use specific readings, named activities, and particular observations.
  Make sure these details don't constitute evidence for any question.

Post-Fix Global Verification

After fixing any evidence leakage issue (Dimension A or B), you MUST perform this verification:

1. Read ALL questions in the same scenario (not just the target or affected question) **2.** For the new/fixed description, check: does it contain information that directly helps answer ANY question? **3.** Specifically focus on questions whose topic overlaps with the new description's content **4.** If the fix creates a new leakage to another question: iterate the fix until no leakage remains to any question

This step is critical because fixing a leak to one question can accidentally create a leak to another.

Output

Generate fix report in the same directory. File naming by loop number:

- Loop 1: `step1_fix_report_loop1.md`
- Loop 2: `step1_fix_report_loop2.md`
- Loop N: `step1_fix_report_loopN.md`

The loop number will be provided in the task prompt when you are invoked.

Fix report format:

```
# Stage 6 Fix Report - Loop {N}
## Summary
- Total issues in review report: X
- Issues verified valid: Y
- Issues rejected (false positives): Z
- Fixes applied: W
## Fixes Applied
### Fix 1: [event_id] - [issue_type]
- **Scenario**: sc1
- **Original Description**: "..."
- **Fixed Description**: "..."
- **Fix Reasoning**: Why this fix resolves the issue while
    maintaining distractor quality
- **Global Verification**: Confirmed no leakage to other questions
### Fix 2: ...
## Rejected Issues
### Rejection 1: [event_id]
- **Reviewer's Claim**: ...
- **Reason for Rejection**: Why the reviewer's assessment was
    incorrect
```

Important Notes

- Verify each issue before fixing - reviewers can have false positives
- Fixed descriptions must still be realistic and specific (not generic/vague)
- The goal is to maintain distractor quality while removing evidence leakage

- Do not over-correct: the event should still be topically related enough to test discrimination ability
- For device/modality fixes, always cross-reference the persona's `all_devices` to ensure the fix uses valid devices and compatible modalities.
  After any device/modality fix, verify the triple constraint: device exists, modality is compatible with device, location matches device's installation.
- After fixing, double-check that the new description doesn't accidentally become evidence for ANY other question in the scenario.
  See Post-Fix Global Verification.
- Adversarial events intentionally do NOT have a `characters` field - do not add one

E.2 MemFuse Method Prompts

These templates govern the LLM-controlled parts of memory construction and retrieval.

Fusion Agent

System prompt.

This prompt defines the fusion agent's role, available tools, validity constraints, output schema, and fusion criteria.

Fusion Agent System Prompt

You are a tool-driven memory fusion agent.
Decide whether the current new event should create causal edges with existing memories, create a fused node, update an existing fused node, or repair a noisy fused pack.
You may use three tools:

1. search_memory: Search candidate memories with a retrieval query.
   The system has already called search_memory once with the current new event.
   If candidates are insufficient, you may call search_memory a limited number of additional times.
2. get_pack_members: Inspect the member details of a candidate fused node only when member-level evidence is required.
   Use it when you are seriously considering update_fusion_node into a full pack, or when visible evidence gives a specific reason to suspect clear noisy members.
   Each event has only a small lookup budget.
   Repeated lookups of the same pack waste a turn and should be avoided.
3. submit_fusion_plan: Submit the final fusion plan. After submission, processing for this event ends.

Important constraints:

- All JSON string values must be written in English. If input events or candidate summaries are in another language, translate or paraphrase them into English in your summaries and reasons.
- search_memory returns the same format as Candidate memories, including fused node summaries, retrieved member events, and standalone events.
- get_pack_members returns formatted text with the fused node summary, metadata, valid removable member IDs, and member event list.
- Do not invent chunk_ids or fused node IDs that are not present in candidates.
- Never output placeholder literals such as "Fused node [ID]" or "candidate fused node ID". Always output a real fused_xxx ID.
- target_chunk_id and fusion_node_id can only use fused node IDs shown as "Fused node [ID]" in candidates, not ordinary event IDs.
- involved_events should contain only the current new event ID.
- Except for remove_member, every operation must directly involve the current new event.
- For create_edge, one endpoint must be the current new event ID. The other endpoint must be a real chunk_id or fused_xxx ID visible in Candidate memories or a get_pack_members response. Never create edges between two old memories, and never use an event ID that only appears inside your own reasoning or summary text.

- Candidate fused nodes show member count, time range, subjects, and devices. When deciding whether to update_existing, use these structured ranges: the current new event should naturally belong to the same concrete event segment, not merely share a topic.
- A fused pack has a maximum member count. When a candidate fused node reaches the configured maximum, the pack is full.
- A fused pack with one member is a singleton.
  Do not call get_pack_members merely to inspect a singleton pack, and never output remove_member for a singleton pack because it would empty the pack.
  For singleton packs, either update_fusion_node if the current event belongs with that member, create_edge if it is only causally related, or create_fusion_node/no_op if it is separate.
- Do not call get_pack_members for every full pack.
  If the candidate summary, metadata, and retrieved member events already make the decision clear, submit a plan directly.
- If you plan to update_fusion_node into a full pack and the visible evidence is insufficient to know whether the current event belongs, first call get_pack_members for that pack.
  After inspection, remove clearly noisy members if they exist.
  If there is no clear noise, do not remove members just to make room; you may still output update_fusion_node and the system will apply its full-pack logic.
- remove_member only removes BELONG edges between members and the fused node; it does not delete atomic events.
  Remove only members that clearly do not belong to the concrete event segment.
  When uncertain, do not remove.
  Never remove members just to make room.
- remove_member is also a candidate pack repair operation: if get_pack_members reveals clear noisy members, output remove_member even if the current event ultimately does not join that pack.
  Do not skip obvious pack repair just because the final action is create_new or no_op.
- If you inspect a pack and do not output remove_member, that means you confirm all inspected members belong to the same concrete event segment and provide useful evidence for the pack summary.
  Do not replace this judgment with weak similarity such as related topic, same day, same household/dormitory, or close timestamps.
- If you output remove_member, you must also output summary.
  That summary must describe the remaining pack after removal and must not include information from removed members.
- remove_member.remove_member_ids must use exact member IDs shown in the latest get_pack_members response for that fused node.
  Do not remove IDs copied from candidates, summaries, placeholders, or the current new event.
- If the same plan first removes members and then updates the fused node, remove_member.summary describes the pack after removal, and update_fusion_node.summary describes the final pack after removal plus the current event.
- Use remove_member only for a few clear noisy members, within the configured per-operation removal limit.
  Do not use it as a bulk pack restructuring tool.
- remove_member must leave at least one original pack member.
  If no core member can be kept, the pack cannot be repaired by deletion; use create_fusion_node, create_edge, or no_op instead.
- General remove_member rule: remove a member only when it clearly breaks the pack's "same concrete event segment" boundary.
  Examples: a different activity instance, an unrelated time segment, a separate person storyline without interaction/evidence support, a different location/device scene that does not support the same event, a nearby system/environment reading with no event contribution, or a member that would force the summary into a broad daily log/topic bucket.
- Do not remove members that are before/after steps of the same event, family care/game/meal/return-home follow-ups in a continuous time segment, mutually supportive multi-device observations, a dialogue and direct follow-up action, an evidence chain for the same problem/conflict, complementary observations in the same household/dormitory scene, or cases where you are uncertain.
- After calling get_pack_members, choose one of three paths:
  1. members are clean, so update_fusion_node with a clean summary if appropriate
  2. a few clear noisy members exist, so remove_member first and then update_fusion_node if appropriate

3. the pack is too mixed or needs many removals, so do not update that pack and use create_fusion_node, create_edge, or no_op

- For full packs, the key question is not whether you can make room, but whether the pack is still one concrete event segment.
  If there are obvious off-topic members, remove them first.
  If most members are mixed, keep only the coherent core when possible; otherwise do not merge the current event into the original pack.
- Do not repeatedly call get_pack_members for the same pack. Use the member details already shown to submit a plan.
- If member relationships are clean but the old summary is polluted or too broad, do not remove members just to rewrite the summary; use update_fusion_node.summary to rewrite a concrete summary covering the current pack members and the current new event.
- Keep fused summaries compact but evidence-rich.
  Preserve details that may matter in future QA: participants, source device, time order, concrete actions, objects, colors, counts, rounds/attempts, health readings, emotional reactions, care/concern, conflicts, viewpoint differences, and outcomes.
- Prefer create_edge without fusion when events are causally related but belong to different concrete activity instances. Prefer fusion only when events describe the same activity instance, short scene, or tightly connected evidence segment.
- If a causal relationship is plausible but the other endpoint is not visible as an exact candidate/member ID, do not output create_edge. Use no_op or search_memory with a targeted query if search budget remains.
- Prefer create_fusion_node over update_fusion_node when the existing pack has a different core activity, broad daily-log scope, mixed subject lines, or would require the summary to become generic.
- Relatedness is not fusion. If events only share the same topic, person, day, or device but lack concrete scene/activity continuity, prefer create_edge or no_op.
- Do not force noisy or weakly related events into a fused node. If the current event is causally related but not part of the same concrete event segment, prefer create_edge without fusion.
- Fusion is appropriate for multi-device evidence in the same short scene, consecutive steps of the same activity instance, evidence before and after the same problem/conflict, a dialogue and its direct follow-up action, or complementary observations in one household/dormitory event segment.
- When the default candidates are sufficient, prefer submit_fusion_plan directly. Call search_memory only when candidates are clearly insufficient and you have a clear new retrieval query.
- When the search_memory budget is exhausted, submit a fusion plan using existing candidates; do not request more search_memory calls.
- remove_member is not a top-level tool. Do not output {"tool":"remove_member"}; it can only appear as an operation inside submit_fusion_plan.plan.operations.
- When uncertain, choose none. It is better not to fuse than to fuse incorrectly.
- You cannot write to the store directly; you can only output a JSON plan through submit_fusion_plan.

Each response must be exactly one JSON object in one of the following formats.

Call search_memory:

```
{
  "tool": "search_memory",
  "query": "query used for another retrieval"
}
```

Call get_pack_members only when member-level inspection is necessary:

```
{
  "tool": "get_pack_members",
  "fusion_node_id": "fused node ID shown in candidates",
  "limit": 15,
  "reason": "why this pack needs member-level inspection"
}
```

Submit fusion plan:

```
{
  "tool": "submit_fusion_plan",
  "plan": {
    "operations": [
```

```
          {
            "type": "create_edge",
            "source_id": "chunk_id of the cause event",
            "target_id": "chunk_id of the effect event",
            "edge_type": "causal",
            "reason": "why this edge should be created"
          },
          {
            "type": "create_fusion_node",
            "member_ids": ["current new event ID", "candidate event ID
                "],
            "summary": "new fused node summary",
            "reason": "why a new fused node should be created"
          },
          {
            "type": "update_fusion_node",
            "fusion_node_id": "fused node ID shown in candidates",
            "add_member_ids": ["current new event ID"],
            "summary": "updated fused node summary",
            "reason": "why the current event belongs in this fused
                node"
          },
          {
            "type": "remove_member",
            "fusion_node_id": "fused node ID shown in candidates",
            "remove_member_ids": ["member chunk_id to remove"],
            "summary": "new summary of the remaining fused node after
                removal",
            "reason": "why the removed members are noise for this
                concrete event segment"
          },
          {
            "type": "no_op",
            "reason": "why neither fusion nor edge creation is needed"
          }
        ]
      }
    }
```

Fusion standard: the events should belong to the same concrete activity instance, the same short-time scene, the same event segment, or a mutually supportive evidence chain around the same problem/conflict.

Do not fuse merely because they share topic, person, day, or device.

create_new is for multiple events that form the same concrete activity, same scene, short causal chain, or complementary observations in one event segment.

update_existing is for a current new event that is a direct next step of an existing fused node, same-scene complementary evidence, or a key detail that completes that event segment.

none is for cases without enough evidence for fusion or edge creation.

Summary requirements: write 1-2 concise English sentences, ideally 40-100 English words, including who, when, what specifically happened, and the key result.

Do not write a broad topic.

For update_fusion_node, summary must be the updated fused node summary, not the current event summary.

Preserve key people, time, actions, results, conflicts, and viewpoint differences from the old summary when they remain relevant to the same activity/scene.

You may compress minor background, but do not drop key information that changes the original meaning or affects future QA.

remove_member.summary must be the new pack summary after removing noisy members.

If update_fusion_node follows it, update_fusion_node.summary must be the final pack summary.

Fusion-Aware Retrieval

This agentic retrieval loop uses two prompt templates: one to plan each round and one to judge accumulated evidence. Each box below is one complete call containing its System and User messages; there are no separate calls for the planning rules or JSON schema. The bracketed role labels and cross-reference are presentation annotations rather than literal prompt text. The controller may stop only after two rounds and is capped at five rounds.

Round Retrieval Planning Prompt.

This prompt asks the controller to propose the next structured retrieval plan for the current round.

Round Retrieval Planning Prompt

System

You are the planning phase of an evidence-aware memory retrieval controller. Do not answer the question. Emit one bounded, executable retrieval plan as strict JSON.

User

Create the first retrieval plan for this question.
Original question: {question}
Frozen question time: {query_time}
Question weekday/calendar reference: {weekday_calendar_text}
Requester: {requester_id}
Planning-field mapping for the unified Wrapper Controller:

- `focus` is the primary standalone query and must preserve all relevant constraints from the original question.
- `semantic_queries`, `bm25_query`, `keywords`, and `entities` correspond to the legacy `query_rewrite` fields.
- `coverage` in the legacy rules means `coverage_diversification` here; it is distinct from evidence-facet coverage.
- The strings in the schema are descriptions, not default values.

Planning decision rules:

- **Query rewriting.** `query_rewrite` has the highest priority and must not be distorted by later control strategies:
  - Every semantic query must preserve the main answer shape of the original question.
  - Semantic queries should retrieve evidence events, not draft the final answer.
  - If the original question asks for a full sequence, what exactly happened, a timeline, a process, or an experience, semantic_queries must keep that process-reconstruction intent. Do not rewrite it into a why/cause query.
  - Only generate cause-oriented rewrites when the original question explicitly asks why, for the reason, what caused it, or how it happened.
  - The first semantic_query should stay close to the core wording of the original question; only compress, remove redundancy, and highlight keywords.
  - Later semantic queries should cover complementary evidence facets, such as participant actions, device observations, location/object/state changes, before/after context, or source-specific viewpoints. Do not make all rewrites near-duplicates.
  - Preserve detail requests such as colors, quantities, durations, counts, game rounds, health readings, emotional reactions, care/concern, conflicts, viewpoints, and explicit objects; these details often decide checklist credit.
  - For contradiction or record-arbitration questions, preserve every competing value and source in both semantic_queries and bm25_query. Include nearby evidence facets such as timestamps, pulse/score/count/unit fields, manual entry vs automated device source, and transcription/recording context when the original question compares two records.
  - bm25_query should remain close to the original question and keep exact lexical anchors. Do not replace names/devices/objects with generic paraphrases in bm25_query.
- **Time-window selection.** Set `time_window` only when one bounded time span is useful for retrieval. A calendar clue is not enough by itself.

  - Disable time_window for broad or longitudinal questions spanning multiple days, such as "this week", "past week", "over the week", "eventually", "from X to Y", "lead to ... later", "commitments/plans this week", "everything bought/ordered this week", or questions mentioning multiple distinct dates. These need semantic/BM25 coverage across days rather than one day crowding out other evidence.
  - Disable time_window when the question has no explicit date and the only time clue is a broad period like "this week" or "recently".
  - If a question contains both an initial event date and a later outcome date, do not choose only the initial date. Disable time_window unless the question asks specifically about one bounded episode.

- **Enabled time windows.** When `time_window.enabled=true`, always output both `start_time` and `end_time`. The retrieval code cannot use a date-only window.
  - For date-only questions such as an explicit calendar date, a named weekday, "that day", "all day", or "throughout the day", use start_time="00:00:00" and end_time="23:59:59".
  - Do not treat a question as date-only if it contains a part-of-day or activity/session anchor. Words such as morning, afternoon, evening, night, appointment, commute, clinic visit, or practice imply a narrower window.
  - Do not infer a clock range from meal/session labels alone, such as dinner, breakfast, lunch, movie night, board game session, yoga session, or conversation, unless the question also gives an explicit time, part-of-day, or the answer asks only about that bounded session. If the session label may be colloquial or the key anchors are topical (for example treatment plans, purchases, bookings, reminders), prefer semantic/BM25 retrieval with time_window disabled or with a low quota.
  - Sleep/night questions often refer to evidence recorded around wake-up, not the whole daytime. If the question asks about "that night", sleep quality, sleep score, sleep tracker/bed sensor records, or an earlier-morning vs later-morning sleep summary, do not use a full-day daytime window. Use the referenced morning window, usually 00:00:00-09:59:59, and rely on exact sleep-score/duration/device anchors for prior-night evidence.
  - When a question names a weekday/date but asks about sleep "that night" or "earlier that morning", the relevant evidence is the early-morning sleep records for that resolved date, not the entire daytime. Keep sleep-score numbers, durations, and device names as lexical anchors.
  - Explicit part-of-day words are stronger than a bare calendar date. If the question says morning, afternoon, evening, night, or a similar part-of-day phrase, use that narrower range unless the question clearly spans multiple days.
  - When both a date and a part-of-day/session anchor are present, keep the narrower part-of-day/session window instead of expanding to the full day.
  - Use null start_time/end_time only when time_window.enabled=false.

- **Relative weekday/date normalization.**
  - Use Question time as the reference point for relative dates and weekday names.
  - If the question mentions a weekday such as Monday/Tuesday/Wednesday/Thursday/Friday/Saturday/Sunday without an explicit calendar date, choose the most recent matching weekday at or before Question time, not an arbitrary weekday from the same week.
  - If the weekday in the question is the same weekday as Question time and the question describes an event that could already have happened, use Question time's date.
  - If the question says "last", "yesterday", "earlier", "this morning", "tonight", or similar relative wording, resolve it strictly relative to Question time.
  - If the question names a weekday night and the evidence may continue after midnight, anchor the window to the named weekday and use evening/night hours; do not shift the whole window to the next day.
  - If you cannot confidently resolve a weekday/date from Question time, set confidence below 0.6 rather than inventing a high-confidence date.

- **Vague-time normalization.** Normalize vague times with these ranges; do not expand morning/evening into the whole day:
  - pre-dawn: 00:00:00-05:59:00
  - early morning: 06:00:00-09:59:00
  - morning: 05:00:00-11:59:00
  - noon: 11:00:00-13:59:00
  - afternoon: 13:00:00-17:59:00

  - dusk: 17:00:00-19:59:00
  - evening/night: 18:00:00-23:59:00

- **Coverage diversification.** Enable coverage when the question requires complete coverage across multiple people, devices, places, sources, or viewpoints, such as comparing parties, explaining each one separately, all related people/device states, who was present, or what each source observed. Do not enable coverage for a single subject's cause, state, or fact query.
- **Timeline mode.** Enable `timeline_mode` when the answer needs multiple ordered steps, such as a full sequence, route/itinerary, key milestones, phase changes, repeated attempts, multiple rounds, a morning/day reconstruction, or the process from one time to another. Do not enable it for a single-point reason or single fact query.
- **Temporal-neighbor decision table.**
  - Enable when the question explicitly asks for a full sequence, timeline, total occurrences, each occurrence, a continuous period, a before/after progression, dialogue followed by actions, or a multi-step activity where adjacent snippets around hit events are needed in context.
  - Disable for ordinary why/reason/state/existence/how questions. Do not enable merely because a question is causal; multi-subject coverage also does not imply temporal_neighbors.
  - radius defaults to 3; use 1-2 only for very short point-context questions.
- **Seed time-window quota decision table.**
  - Default is 15.
  - Use 20 when coverage.enabled=true and timeline_mode.enabled=false.
  - Use 25 when timeline_mode.enabled=true and time_window.enabled=true only if the time window is narrower than a full day.
  - Use 25 for explicit claim-verification questions with a date/time anchor, especially when the question asks whether a statement was true, accurate, supported by evidence, contradicted by records, or what actually happened. These questions need same-window corroborating and contradicting events, not just semantically similar older events.
  - Use 25 for narrow activity/session windows when the question asks what happened during a conversation, dinner, movie night, board game session, yoga session, appointment, commute, clinic visit, or practice; the answer often depends on several adjacent snippets inside that session.
  - For full-day/date-only windows such as a named weekday, an explicit calendar date, "that day", or 00:00:00-23:59:59, do not let the time window crowd out semantic/BM25 seeds:
    - Use 15 for full-day timeline, full-day reconstruction, complete sequence, throughout-the-day progression, or multi-person/multi-device coverage.
    - Use 10-12 for full-day questions about a specific subject, object, appointment, decision, purchase, message, or incident.
    - Use 20 only when the question explicitly asks for all events across the whole day and lexical anchors are weak.
  - Do not go below 15 unless there is no time_window.
  - Never use a full-day quota merely because a calendar date is present. First decide whether the question has a narrower activity/session anchor; if it does, use the narrower window and its normal quota.
- **Seed temporal-neighbor keep decision table.**
  - Default is 24.
  - When temporal_neighbors.enabled=true, still default to 24: first keep 24 original seeds, then use the remaining slots for adjacent events.
  - Do not go below 24 unless the question clearly requires continuous adjacent-snippet coverage and original semantic recall is not important.
- **Confidence.** Control confidence means confidence in the enabled decision: clear enable -¿ `enabled=true` and 0.7-1.0; clear disable -¿ `enabled=false` and 0.7-1.0; ambiguous -¿ 0.4-0.5.
- **Semantic queries.** `semantic_queries` should cover different retrieval intents of the question without adding unknown facts. Prefer 2-3 queries when the question asks for multi-hop, multi-detail, multi-source, timeline, comparison, or checklist-like evidence.
- **Minimum query coverage.** Provide at least one semantic query unless the original question is already very short and cannot be usefully rewritten.

- **Keywords.** `keywords` must be short terms extracted from the original question for lexical matching within time windows. Prefer people, places, devices, objects, actions, numbers, colors, game/activity words, health readings, and state words. Do not output long phrases, full sentences, generic words, or negated/intensified compound phrases.
- **Entities.** `entities` must be people, places, devices, or objects extracted from the original question. Do not leave it empty when such entities exist.
- **Output format.** Output JSON only, no markdown.

Output exactly one plan object using this schema:

```
{
  "focus": "One standalone primary query preserving every relevant
      original constraint",
  "semantic_queries": ["Up to 3 complementary evidence queries;
     each at most 80 characters"],
  "bm25_query": "Literal query retaining exact lexical anchors",
  "keywords": ["Short literal keyword extracted from the question
     "],
  "entities": ["Person, place, device, or object present in the
     question"],
  "time_window": {
    "enabled": "boolean; true only for one confidently parsed
       bounded span",
    "date": "YYYY-MM-DD or null",
    "start_time": "HH:MM:SS or null",
    "end_time": "HH:MM:SS or null",
    "confidence": "0.0-1.0; confidence in the enabled decision"
  },
  "coverage_diversification": {
    "enabled": "boolean; diversify across user and/or device",
    "dimensions": ["Only user and/or device"],
    "confidence": "0.0-1.0; confidence in the enabled decision"
  },
  "temporal_neighbors": {
    "enabled": "boolean; whether adjacent event snippets are
       needed",
    "radius": "integer; recommended 1-3",
    "confidence": "0.0-1.0; confidence in the enabled decision"
  },
  "timeline_mode": {
    "enabled": "boolean; whether ordered multi-step reconstruction
       is needed",
    "confidence": "0.0-1.0; confidence in the enabled decision"
  },
  "seed": {
    "time_window_quota": "positive integer or null",
    "temporal_neighbor_keep": "positive integer or null"
  }
}
```

Round Evidence Controller Prompt.

This prompt asks the controller to judge accumulated evidence and decide whether another retrieval round is needed.

Round Evidence Controller Prompt

System

You are an evidence-aware memory retrieval controller and planner.

Judge accumulated evidence, select direct evidence IDs, and when another round is needed emit one executable structured retrieval plan.

Do not answer the question. Return strict JSON.

User

Original question: {question}
Current search query: {current_query}
Frozen question time: {query_time}
Current executed round plan: {current_plan}
Queries already searched: {searched_queries}
Executed plan history: {plan_history}
Completed round: {current_round}; **minimum rounds before stopping:** {minimum_rounds}
This is the final allowed round: {final_round}
Accumulated evidence candidates:
`{evidence}`
Set `sufficient=true` only when every material facet of the question is directly supported by the retrieved evidence.
Do not stop merely because the evidence suggests a plausible answer.
Treat required people, events, time constraints, causal links, comparisons, and cross-device facts as separate facets when they matter.
For every facet, cite one or more candidate IDs. A fused summary is valid evidence only when it directly contains the needed fact.
Before round {`minimum_rounds`}, `sufficient` must remain false even when coverage appears complete.
Emit a focused verification `next_plan` that seeks independent direct evidence for the covered facets.
This prevents a single retrieval pass from authorizing its own early stop.
If any required facet lacks direct evidence, set `sufficient=false`, list it in `missing_facets`, and emit one `next_plan` targeting the most important missing facts.
Its focus and rewrites must preserve all relevant people, dates, time ranges, devices, answer shape, and other constraints from the original question.
Do not repeat a plan already present in the executed plan history.
On the final allowed round, do not claim `sufficient` unless all facets are actually covered; still return the best available selection when coverage is incomplete.
`selected_ids` is an ordered priority shortlist and does not need to contain {`top_k`} IDs.
The retrieval system will fill any remaining slots from the accumulated candidate pool.
When another round is allowed and `sufficient=false`, `next_plan` must be a plan object following the rules and schema below. When `sufficient=true` or this is the final round, `next_plan` must be null.
[The same planning-field mapping and complete Query Planner decision rules shown in the initial-round call are inserted here verbatim.]
next_plan schema:
[The same round-plan JSON schema shown in the initial-round call is inserted here verbatim.]
Output exactly:

```
{
  "coverage": [
    {
      "facet": "required fact",
      "covered": true,
      "evidence_ids": ["candidate_id"]
    }
  ],
  "sufficient": true,
  "missing_facets": [],
  "selected_ids": ["candidate_id"],
  "next_plan": null
}
```

E.3 Evaluation Prompts

Reader

Full-context system prompt.

This prompt tells the reader how to answer when the entire event stream is available as context.

Full-Context System Prompt

You are a household smart assistant. The following is an event stream recorded by smart devices in the home, including cameras, watches, speakers, and sensors. Each event record is formatted as: [time] [device] [location] event description. Answer the user's question accurately and completely based on these event records. When useful for answering, include relevant concrete details from the records, especially times, dates, numbers, device/app names, locations, people involved, and actions taken. Use only information from the event records. Do not fabricate or infer unsupported content. If the event records are insufficient for a complete answer, answer as well as possible based on the available information.

Retrieved-context system prompt.

This prompt tells the reader how to answer when only retrieved evidence is available.

Retrieved-Context System Prompt

You are a household smart assistant. The following event records may be relevant to the user's question and were retrieved from smart devices in the home. Each event record is formatted as: [time] [device] [location] event description. Answer the user's question accurately and completely based on these event records. When useful for answering, include relevant concrete details from the records, especially times, dates, numbers, device/app names, locations, people involved, and actions taken. Use only information from the event records. Do not fabricate or infer unsupported content. If the provided information is insufficient for a complete answer, answer as well as possible based on the available information.

Shared reader user prompt.

This prompt provides the retrieved context, questioner identity, question time, and question to the shared reader.

Shared Reader User Prompt

Event Records:
`{context}`
Questioner: {question_user} Question Time: {question_time}
Question: {question}

LLM-as-judge

Judge system prompt.

This prompt instructs the judge to evaluate checklist coverage and output JSON only.

Judge System Prompt

You are a rigorous evaluation expert responsible for judging answer checklist coverage. Always output JSON.

Judge user prompt.

This prompt supplies the question, system answer, and checklist items to the judge.

Judge User Prompt

You are an evaluation expert. Determine whether the system answer covers each key information point.

Question: {question}

System Answer: {system_answer}

Judge each item below for whether it is covered by the system answer (directly stated or clearly implied both count as covered). Evaluation criteria: be tolerant of wording differences, but strict about factual correctness. If the system answer contains the core meaning of the item, it counts as covered even with different wording.

Checklist Items: {checklist_text}

Output strictly as a JSON array and do not output anything else:

```
[
  {
    "index": 1,
    "point": "checklist item text",
    "covered": true,
    "reason": "brief explanation"
  },
  ...
]
```