ONE ASSISTANT, MANY MEMORIES: BENCHMARKING REAL-WORLD GROUP ASSISTANCE WHEN RECALL IS NOT ENOUGH

Anonymous authors
Paper under double-blind review

ABSTRACT

The value of memory lies not only in recall but in using it to provide appropriate assistance. Recent benchmarks test memory use in personalisation and agentic tasks, but seldom evaluate assistance without fixed answers or integration across multiple participants. Group chats make both challenges more acute: an assistant must integrate information from multiple members, track revisions and conflicts over time, and produce one response for the group. Yet existing group memory benchmarks rely on synthetic conversations and inherit constrained evaluation from general memory benchmarks, focusing on recall against fixed answers or decisions over a closed set. We introduce GROUPASSISTBENCH, the first memory benchmark from real-world group conversations with a deployed group assistant, covering 80,299 messages in 50 groups and costing over $53K.[1] Over the same histories, GROUPASSISTBENCH evaluates memory recall (L1) with reference answers and memory use (L2) with service requests without fixed answers across single, distributed, and conflicting memories (M1–M3). Requests are verified to require group history, memories are anchored to their sources, and L2 uses tailored rubrics with penalties derived from real user feedback. Across evaluations of frontier models with full context, retrieval methods, and memory systems, the strongest memory system rivals full context on recall but falls far behind on use, especially for distributed and conflicting memories. Together, two additions that relate retrieved memories and bind remembered facts as response constraints bring consistent gains, showing that group memory systems must improve use, not recall alone.

1 INTRODUCTION

Memory is a core component of agents built on LLMs and has been widely studied in recent years (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2025). Recent benchmarks have moved beyond factual recall to evaluate whether remembered information supports personalization (Jiang et al., 2025; Xiao et al., 2026; Xie et al., 2026) and agentic task completion (He et al., 2026; Zhao et al., 2026). This shift from remembering information to using it effectively becomes especially challenging in multi-user settings. Group chats provide a particularly demanding setting for this shift from recall to use: unlike a personal assistant serving one user, an assistant in a group chat serves the group as a whole and must produce a single response that reconciles information distributed across members, revised over time, or contested within the group (Yang et al., 2026; Hu et al., 2026b; Owolabi, 2026; Ren et al., 2026).

Existing benchmarks for group memory already examine how information is attributed, integrated, and updated, and show that systems designed for one user exhibit distinct failures in group settings (Yang et al., 2026; Owolabi, 2026). Yet their evaluation inherits a broader limitation of current memory benchmarks: success is measured primarily through recall. Questions ask who said what or which value is current and are scored against fixed reference answers, while tasks closer to assistance use multiple choice (Hu et al., 2026b) or selection from a closed set of actions (Ren et al., 2026). This leaves unanswered whether a system that retrieves the right memory can use it to respond appropriately to a service request without a fixed answer. Evidence from settings with one user shows why this distinction matters: on AlpsBench, memory raises persona awareness from 0.41 to 0.72 while reducing emotional appropriateness below the backbone without memory (Xiao et al., 2026). A separate limitation is that current benchmarks for group memory use synthetic conversations, leaving it unclear whether their tasks capture how agreements, corrections, and assistance failures arise in deployed groups (Owolabi, 2026; Hu et al., 2026b).

In this paper, we introduce GROUPASSISTBENCH, a benchmark built on 50 group conversations involving a deployed assistant, containing 80K messages and 2,395 questions. GROUPASSISTBENCH organises evaluation along two orthogonal axes, include: 1) The task axis, aims to distinguish memory recall (L1), which asks a question about the

[1] See Appendix I for the cost breakdown.

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.

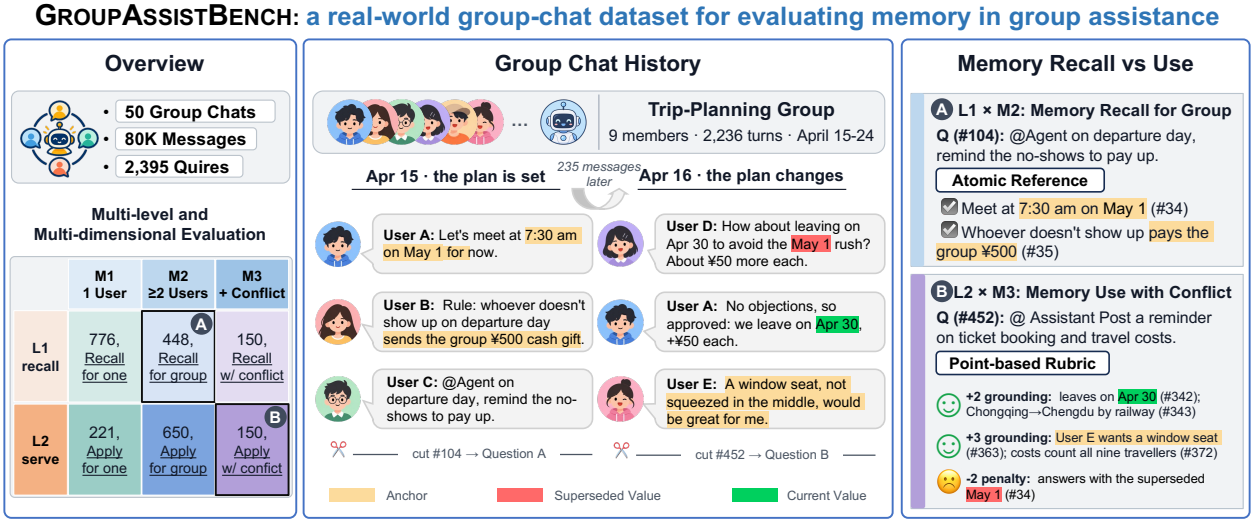


Figure 1: Overview of GROUPASSISTBENCH. **Left:** two task levels, memory recall (L1) and memory use (L2), crossed with three types of memory dependency (M1–M3). **Middle:** a thread from a trip-planning group. **Right:** a recall question and a use request derived from the same thread with different history cutoffs: the fact that answers the former is a superseded value for the latter. Messages are translated and abridged.

history and has a reference answer, from memory use (L2), which poses a service request without a fixed answer and requires a reply that respects the memories it depends on. This axis measures whether a model can move from retrieving remembered information to using it appropriately in an open-ended response, and 2) The dependency axis, aims to distinguish whether the required memories are single (M1), distributed across members (M2), or conflicting (M3). This axis measures how well a model can integrate and reconcile memory evidence as its dependency structure becomes more complex. Figure 1 gives an overview of GROUPASSISTBENCH. Each request is verified to require the group history, and each required memory is anchored to a verbatim span in the original conversation. L1 answers are scored on atomic facts, while L2 replies are scored with rubrics tailored to each instance, whose penalty criteria are derived from real user feedback on the assistant. As shown in Table 1, GROUPASSISTBENCH is the first benchmark that combines conversations among multiple participants with a deployed assistant, memory use through requests without fixed answers, and criteria derived from feedback.

We evaluate 14 configurations on GROUPASSISTBENCH, covering frontier LLMs with full context (e.g., Claude-sonnet-5 and GPT-5.6-terra), retrieval-augmented systems, and external memory systems. The results reveal a gap along each axis. Along the task axis, the strongest memory system surpasses Full Context with the same backbone on memory recall (0.621 vs. 0.570) but falls far behind on memory use (0.247 vs. 0.431). Along the dependency axis, the Full Context score for memory use drops from 0.505 on M1 to 0.303 on M3. Further analysis highlights two promising directions for group memory assistants: relational evidence organisation and assistance constraint binding (§5). Together, these results show that evaluating recall alone overstates the value of memory to a group. Our contributions are threefold:

1. We introduce GROUPASSISTBENCH, the first benchmark built on real group conversations to evaluate memory along two axes: recall versus use, and single versus multiple or conflicting memories.
2. We develop a traceable framework that verifies history dependence, grounds memories in source messages, and evaluates responses using criteria specific to each request and based on real user feedback.
3. Experiments across full context, retrieval methods, and memory systems reveal complementary retrieval and application bottlenecks that relational evidence organisation and assistance constraint binding mitigate.

2 RELATED WORK

Memory Evaluation from Recall to Use. Long-term memory benchmarks began with factual recall over extended conversation histories (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2026c) and have since turned to how memory supports downstream behavior, such as personalized responses (Jiang et al., 2025; Xiao et al., 2026; Xie et al., 2026),

Table 1: Comparison of GROUPASSISTBENCH with existing memory benchmarks. *Open service* denotes generation for service requests without fixed answers. *Rubric* and *Feedback* indicate criteria tailored to each instance and criteria derived from real user feedback. *#Eval.*is the number of evaluated questions or checkpoints. ○ denotes partial coverage, including constrained outputs such as multiple choice, template filling, or a fixed action set.

Benchmark	Group	Source	Update or conflict	Open service	Rubric	Feedback	#Eval.
Memory for one user and agent tasks							
MemoryArena (He et al., 2026)	✗	partly synthetic	✗	○	✗	✗	766
AMA-Bench (Zhao et al., 2026)	✗	agent trajectories	✓	✗	✗	✗	2,496
PersonaMem-v2 (Jiang et al., 2025)	✗	synthetic	✓	○	✗	✗	5,000
AlpsBench (Xiao et al., 2026)	✗	real user logs	○	✓	✗	✗	2,500
DynamicMem (Xie et al., 2026)	✗	synthetic	✓	○	✗	✗	3,634
Memory for groups							
GroupMemBench (Yang et al., 2026)	✓	synthetic	✓	✗	✗	✗	715
EverMemBench (Hu et al., 2026b)	✓	synthetic	✓	○	✗	✗	2,400
SocialMemBench (Owolabi, 2026)	✓	synthetic	✓	✗	✗	✗	1,031
GateMem (Ren et al., 2026)	✓	synthetic	○	○	✓	✗	2,218
GROUPASSISTBENCH (ours)	✓	real group chats	✓	✓	✓	✓	2,395

implicit constraints on later replies (Li et al., 2026), and multi-session agentic tasks (He et al., 2026; Zhao et al., 2026). All of them target a single user and score memory use in a closed form, or judge free-form answers holistically (Wu et al., 2025; Li et al., 2026), which does not tell which memory was used or misused. Rubric-based evaluation is finer grained (Arora et al., 2025; Ding, 2026), but its criteria are written by benchmark designers, and real user feedback has served only as a satisfaction label on the whole response (Hong et al., 2026). GROUPASSISTBENCH scores open-ended memory use whose required memories come from multiple users, with grounding criteria instantiated from the verified memories of each request and penalty criteria derived from real user feedback on the assistant's failures, so that the criteria name which memory was misused.

Memory in Multi-Party Conversations. Recent benchmarks extend memory evaluation to groups, covering speaker-grounded belief tracking and group dynamics (Yang et al., 2026), temporally evolving decisions across interleaved groups (Hu et al., 2026b), group norms versus individual exceptions (Owolabi, 2026), and memory governance among multiple principals (Ren et al., 2026). They show that memory systems designed for a single user fail characteristically in groups, for example by conflating speakers or overwriting earlier statements. As Table 1 summarizes, their conversations are LLM-synthesized and their tasks are scored against fixed answers, with assistance-like tasks cast as multiple-choice questions (Hu et al., 2026b) or a closed set of actions (Ren et al., 2026); where per-instance criteria exist, they specify required content (Ren et al., 2026) or penalize misattribution (Owolabi, 2026) by design rather than from observed user reactions. GROUPASSISTBENCH instead builds on conversations with a deployed assistant and evaluates recall and use over the same histories.

3 GROUPASSISTBENCH

GROUPASSISTBENCH evaluates whether a system can recall a group's history and, more importantly, whether it can use that history to assist the group. This section formulates the tasks (Section 3.1), presents the construction pipeline that turns group conversations into instances (Section 3.2), and defines the scoring of responses (Section 3.3).

3.1 TASK FORMULATION

A group conversation is a chronologically ordered sequence $H = (m_1, \ldots, m_T)$, where each message $m_t = (u_t, \tau_t, x_t)$ consists of a speaker u_t, who is either a group member or the assistant, a timestamp τ_t, and a text x_t. An instance of GROUPASSISTBENCH is a tuple (c, q, A, K). The cutoff c specifies the history $H_{<c}$ that a system may access, and q is a query posed at the cutoff. $A = \{a_1, \ldots, a_n\} \subset H_{<c}$ is the set of *required memories*, the facts established in the history that a good response to q depends on, each anchored to a message in $H_{<c}$. K is the answer key, a set of reference facts for memory recall or a rubric for memory use. Given $H_{<c}$ and q, a system produces a

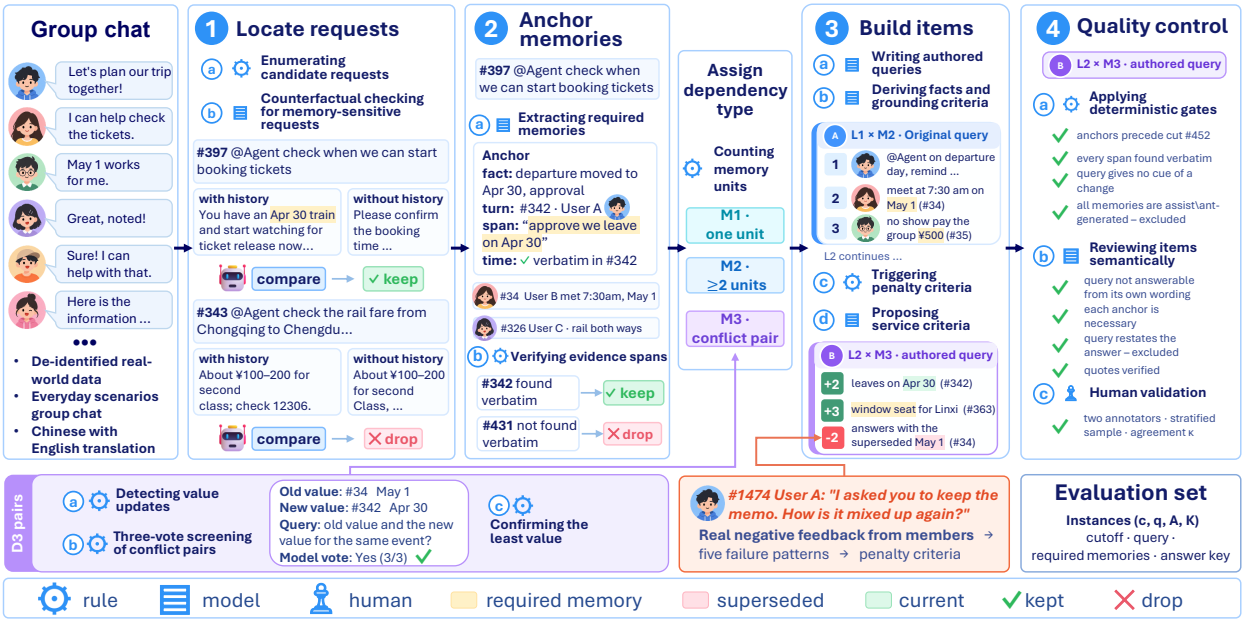


Figure 2: Construction pipeline of GROUPASSISTBENCH from group conversations to evaluation instances. Requests are retained only when the group history changes the answer. Required memories are anchored to verbatim spans, and their structure determines the dependency type. Instances use reference facts for L1 or rubrics for L2, with penalties derived from real user feedback, before passing deterministic gates and semantic review. Icons indicate whether each step uses rules, a language model, or humans. Appendix A provides details.

response $\hat{y}$, and its goal is to score as high as possible under K. A is never shown to the system: it determines the dependency type of the instance, generates K, and allows the evidence a system retrieves to be checked against the memories it should have found. Every instance is labeled along two orthogonal axes, the *task level* (L1 or L2), which fixes the form of q and K, and the *memory dependency* (M1 to M3), which describes the structure of A.

Task axis. In *memory recall* (L1), q is a question about the history itself, such as which restaurant the group has settled on, and K is a set of reference facts, one for each memory in A. In *memory use* (L2), q is an open-ended request for assistance, such as drafting a plan, making a recommendation, or writing a summary for the group, and K is a rubric specific to the instance, since the memories in A constrain an appropriate response without determining a single output. For the reminder request in Figure 1, for example, the rubric checks that the reply uses the revised departure date and counts all nine travellers, and penalizes reverting to the original date. Both levels are built over the same group histories, which allows us to compare how well a system recalls the memories with how well it uses them.

Memory axis. *Single memory* (M1) instances depend on a single substantive memory. *Multiple memories* (M2) instances depend on multiple memories that must be combined, which in a group are typically contributed by different members. *Conflicting memories* (M3) instances have the structure of M2 and additionally contain a *conflict pair* $(a^{\text{old}}, a^{\text{new}}) \subset A$: two values of the same matter, stated at different times or by different members, of which the conversation has made a^{new} the current one. The query gives no cue that anything has changed, so a system must notice the revision on its own; the departure date of Figure 1, moved from May 1 to April 30, is such a pair. These types describe the structure of memory dependency rather than a guaranteed order of difficulty.

3.2 BENCHMARK CONSTRUCTION

The construction pipeline of GROUPASSISTBENCH is shown in Figure 2. We *locate* the requests whose answer depends on the group history, *anchor* the memories each request requires to the messages that establish them, *assign* the dependency type from the structure of the anchors, *build* the instance with its answer key, and apply *quality control*. Throughout, every field produced by a language model is checked programmatically before it enters the benchmark, and instances that fail are excluded rather than repaired.

Data source and statistics. GROUPASSISTBENCH is built from Chinese group conversations in a deployed application, and the assistant replies are its original outputs. Messages are pseudonymized and rewritten to remove residual personal information, with English translations provided. The groups span everyday settings such as trip planning, classes, hobby communities, and project coordination. The evaluation set contains 2,395 instances sampled from 9.5K validated candidates across 50 groups and 80K messages. Although M2 is defined by the number of required memories, 56% of M2 and 93% of M3 instances draw anchors from at least two members. Table 2 reports the full statistics. Appendix A.1 details source selection, and Appendix G provides more examples.

Locating requests. A request enters the benchmark only if answering it requires the group history. We enumerate the messages that address the assistant, discard those that are not requests, and answer each remaining request q posed at message t_q twice with the same model, with and without the preceding history, $\hat{y}^+ = f(H_{<t_q}, q)$ and $\hat{y}^- = f(\varnothing, q)$. A comparator keeps q only if $\hat{y}^+$ contains group-specific information that $\hat{y}^-$ cannot supply and that materially improves the response. A retained request becomes the query of a *native* instance, with the cutoff $c = t_q$.

Anchoring memories. For each retained request we identify the minimal set of memories needed to respond. Each memory is recorded as an *anchor* $a = (f, t, e)$: a self-contained statement of the fact f, the message t that establishes it, and an evidence span $e \sqsubseteq x_t$, a verbatim substring of that message. An extractor proposes anchors and drops any whose removal would not change the response. The verified anchors of all requests form a per-group inventory, from which authored instances are later built.

Assigning dependency types. The dependency type describes the structure of A. Anchors from the same message and near-duplicate facts are merged into memory units; an instance is M1 if it depends on one unit and M2 if on two or more. M3 additionally requires a verified conflict pair $(a^{\text{old}}, a^{\text{new}}) \subset A$. Candidate pairs are detected from update expressions and value changes in the history and matched to the anchor of the earlier value. Since two similar values may concern different matters, e.g., a lodging budget and a ticket price, each candidate receives three independent votes on whether the values concern the same matter, and only unanimously accepted pairs are used. Finally, we verify that a^{new} is still current at the cutoff, and move the cutoff before the next change of the same matter if it is not.

Building instances. Native requests cover only the memories that members happened to ask about. To cover the remaining anchors, we also build *authored* instances, in which a query is written for a set of anchors A drawn from the inventory. Every factual premise in the query must cite a message t and a span $e \sqsubseteq x_t$, verified as for anchors, and the cutoff is placed between $\delta_{\min}$ and $\delta_{\max}$ messages after the last anchor, with no update of the same matter in between. The answer key is then derived from A. For L1, K is the set of reference facts $\{f_a : a \in A\}$, one per anchor. For L2, K is a rubric assembled from three kinds of criteria,

$$K \;=\; R_{\text{ground}}(A) \;\cup\; R_{\text{pen}}(A) \;\cup\; R_{\text{serv}}, \tag{1}$$

where grounding criteria are instantiated from the anchors, penalty criteria are added by deterministic triggers over the anchors and the memory state, and service criteria are proposed by a language model for the request itself.

Quality control. Every instance passes deterministic gates: each anchor and cited premise must precede the cutoff, each span must be found verbatim in its message, and the query must neither reveal a required memory nor, for M3, hint that a value has changed. The instances selected for evaluation additionally pass a language-model review, which checks whether each anchor is necessary, whether the query can be answered from its own wording, whether an anchor in an assistant message relays a fact established by a member, and, for M3, whether the two values concern the same matter; every negative judgment must quote a span that supports it, which is again verified programmatically (Appendix A.6). To assess the resulting quality, three annotators independently examined 120 instances stratified by level, dependency, and source: 95.8% are valid by majority vote and 84.2% unanimously, with pairwise agreement of 93–98% per question. On the 60 L2 instances, the annotators further rated each rubric for the soundness of its service criteria, the relevance of its penalty criteria, and its overall completeness; 90.0% of the rubrics pass all three checks by majority and 86.7% unanimously (Appendix C).

3.3 EVALUATION

The two levels are scored with different units of judgment, atomic facts for memory recall and rubric criteria for memory use, so that failures of recalling a memory can be separated from failures of using it.

Memory recall. For an L1 instance i, the answer key K_i is a set of reference facts G_i. A judge model decides for each fact $g \in G_i$ whether the response $\hat{y}_i$ recovers it, up to paraphrase; let $m_{ig} \in \{0, 1\}$ denote this judgment. Atom

Table 2: Statistics of GROUPASSISTBENCH. **Left:** number of instances by task level and memory dependency. **Right:** statistics over the 50 groups.

	M1	M2	M3	Total
L1: memory recall	776	448	150	1,374
L2: memory use	221	650	150	1,021
Total	997	1,098	300	2,395

	Total	Min.	Mean	Max.
Instances per group	2,395	4	47.9	110
Messages per group	80,299	68	1,606	4,442
Context tokens	4.92M	2.9K	98.5K	972.8K
Anchors per instance	6,322	1	2.64	59

Recall is the fraction of reference facts recovered,

$$R_i = \frac{1}{|G_i|} \sum_{g \in G_i} m_{ig}. \tag{2}$$

Let $H_i = \sum_{g \in G_i} m_{ig}$ be the number of recovered facts. The judge also counts additional factual points E_i, yielding factual precision $P_i = H_i/(H_i + E_i)$ and $F_{1,i} = 2P_iR_i/(P_i + R_i)$.

Memory use. An L2 response is judged against its rubric K_i (Equation 1). A grounding criterion is met only if the response explicitly reflects the fact of its anchor. Penalty criteria describe failures of memory use that we summarised from negative feedback that members gave to the assistant: distorting a recalled fact, contradicting a settled plan, violating an explicit agreement or a member's constraint, misreading a member's current state, and relying on a superseded value; a penalty criterion is met when the response commits the failure. A judge evaluates each criterion independently. Positive criteria receive weights from 1 to 3, with the largest for required memories, while penalties receive weights from -1 to -3. For binary judgment m_{ij} and signed weight w_{ij} on criterion j, the rubric score is

$$S_i = \frac{\sum_j w_{ij}\, m_{ij}}{\sum_{j:\, w_{ij}>0} w_{ij}}. \tag{3}$$

Criteria and reference facts are shown only to the judge; weights and trigger rules are given in Appendix B.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Baselines. Following prior work Zhao et al. (2026); He et al. (2026), we compare three memory-access paradigms: *Full Context*, *Retrieval-Augmented Systems*, and *External Memory Systems*. We characterise their memory organisation using the topology taxonomy of Hu et al. (2025): 1D *flat* memory has no explicit inter-unit relations, 2D *planar* memory organises units within a single structural layer, and 3D *hierarchical* memory connects representations across multiple abstraction levels. Full Context is a 1D reference that directly provides all prior messages and assistant trajectories, and we evaluate four backbones: Claude-sonnet-5, GPT-5.6-terra, Deepseek-v4-flash and Qwen-3.5-9b. Retrieval-Augmented Systems index the same history and include two 1D methods, BM25 and dense embedding retrieval, one 2D method, GraphRAG Edge et al. (2024), and one 3D method, RAPTOR Sarthi et al. (2024). External Memory Systems abstract, organise, or update the interaction history before retrieval. They include two representatives at each level: Mem0 Chhikara et al. (2025) and LightMem Fang et al. (2026) for 1D, A-MEM Xu et al. (2025) and MemoryOS Kang et al. (2025) for 2D, and xMemory Hu et al. (2026d) and EverMemOS Hu et al. (2026a) for 3D. This design covers both retrieval and external memory across the three topology levels without treating structural complexity as an ordering of capability.

Implementation details. We use GPT-5.6-terra as the task agent for retrieval and external memory systems and GPT-5.6-sol as the evaluator. BM25 and dense retrieval return the top 20 units. All embedding matching uses `text-embedding-3-small`. Because the source conversations are in Chinese, we execute and score tasks in Chinese to preserve their meaning and context and avoid translation artefacts. We provide a complete English translation for reference and future research. Each question has a cutoff that limits context and memory construction to earlier turns, preventing answer leakage. Appendices E and H detail the setup and prompts.

Evaluation metrics. Using GPT-5.6-sol as the judge, we report mean Atom Recall and factual F1 for L1, mean rubric score $\bar{S}$ for L2, and mean context tokens. We break results down by M1–M3 and rank methods by overall

Table 3: Main results on GROUPASSISTBENCH for memory recall (L1) and memory use (L2). All RAG and external memory systems use GPT-5.6-terra as the task agent. We report Atom Recall (met) and F1 for L1. For L2, $\bar{S}$ denotes the mean of the rubric scores S_i. Ranks are based on mean met over all L1 questions and mean S_i over all L2 questions. Bold numerical entries indicate maximum scores across all methods and rank 1. Underlined numerical entries indicate maximum scores and the highest rank within each group.

Method	Memory type	L1-M1		L1-M2		L1-M3		L1 avg.	L1	L2-M1	L2-M2	L2-M3	L2 avg.	L2	Ctx.
		met ↑	F1 ↑	met ↑	F1 ↑	met ↑	F1 ↑	met ↑	rank ↓	$\bar{S}$ ↑	$\bar{S}$ ↑	$\bar{S}$ ↑	$\bar{S}$ ↑	rank ↓	tokens
Full Context															
Claude Sonnet 5	1D	0.739	0.570	0.504	0.460	0.429	0.352	0.628	2	0.473	0.426	0.371	0.428	3	52,435
DeepSeek-v4-flash	1D	**0.766**	0.526	**0.572**	**0.472**	0.455	0.353	**0.669**	**1**	**0.510**	**0.478**	**0.374**	**0.469**	**1**	52,435
Qwen-3.5-9b	1D	0.622	0.467	0.344	0.341	0.285	0.278	0.494	11	0.446	0.343	0.226	0.348	6	52,435
GPT-5.6-terra	1D	0.685	**0.623**	0.436	0.453	0.377	**0.356**	0.570	4	0.505	0.435	0.303	0.431	2	52,435
Retrieval-Augmented Systems															
BM25	1D	0.676	0.612	0.378	0.404	0.313	0.306	0.539	6	0.446	0.376	0.230	0.370	4	10,769
Dense retrieval	1D	0.651	0.595	0.374	0.397	0.358	0.351	0.529	7	0.428	0.368	0.242	0.362	5	9,539
GraphRAG	2D	0.618	0.554	0.356	0.384	0.356	0.343	0.504	10	0.399	0.351	0.236	0.344	7	18,468
RAPTOR	3D	0.647	0.581	0.350	0.376	0.345	0.329	0.517	8	0.422	0.327	0.237	0.334	8	14,151
External Memory Systems															
Mem0	1D	0.464	0.408	0.252	0.264	0.234	0.223	0.370	13	0.305	0.241	0.172	0.245	11	1,190
LightMem	1D	0.290	0.261	0.133	0.155	0.178	0.189	0.227	14	0.192	0.131	0.094	0.139	14	742
A-MEM	2D	0.537	0.406	0.342	0.300	0.355	0.261	0.454	12	0.321	0.218	0.185	0.235	12	5,387
MemoryOS	2D	0.623	0.451	0.387	0.327	0.327	0.234	0.513	9	0.380	0.255	0.168	0.269	9	5,203
xMemory	3D	0.620	0.511	0.455	0.374	0.435	0.262	0.546	5	0.133	0.223	0.220	0.203	13	5,841
EverMemOS	3D	0.679	0.555	0.540	0.430	**0.561**	0.302	0.621	3	0.137	0.284	0.253	0.247	10	21,123

Atom Recall and $\bar{S}$, following Section 3.3. To validate the judge, three annotators independently marked every rubric criterion for 180 real responses to 60 sampled L2 instances. The judge agrees with the human majority on 89.3% of the 1,152 criterion judgments, the same level as the annotators agree with one another (89.8%). When errors occur, the judge tends to give too little credit to grounding and service criteria and apply penalties too often, making scores conservative. The judge also ranks the three systems identically to every annotator. Appendix C.3 provides details.

4.2 MAIN EXPERIMENT

Overall results. Table 3 shows that Full Context methods remain competitive on both L1 and L2 tasks. DeepSeek-v4-flash obtains L1 met of 0.669 and L2 rubric score $\bar{S}$ of 0.469, while GPT-5.6-terra reaches 0.570 and 0.431, respectively. Within RAG systems, BM25 and dense retrieval obtain higher overall scores than GraphRAG and RAPTOR on both tasks, indicating that additional structural complexity does not consistently improve performance in this setting. Among external memory systems, hierarchical methods show promising L1 recall: EverMemOS and xMemory attain the met of 0.621 and 0.546, placing third and fifth among all evaluated baselines. Flat retrieval is more effective within RAG, whereas hierarchical systems show stronger L1 recall within external memory. This pattern is consistent with concerns that summary hierarchies can obscure factual distinctions Hu et al. (2026d) and highlights hierarchical memory organisation as an emerging direction for supporting reasoning across abstraction levels Hu et al. (2025).

Performance across memory dependencies. Performance generally degrades as queries move beyond dependence on a single memory. L1 met decreases from M1 to both M2 and M3 for all 14 evaluated baselines, and L2 shows a similar pattern except for xMemory and EverMemOS. For example, with GPT-5.6-terra under Full Context, L1 met drops from 0.685 on M1 to 0.436 and 0.377 on M2 and M3, while the L2 score decreases from 0.505 to 0.435 and 0.303. These drops highlight the difficulty of integrating complementary memories across speakers or turns in M2 and selecting the current or applicable memories amid updates, conflicts, or contextual constraints in M3. The two 3D external memory systems, EverMemOS and xMemory, show the strongest L1 robustness under this shift, with smaller M1 to M3 declines than every Full Context and RAG configuration. EverMemOS declines from 0.679 to 0.561, while Full Context with GPT-5.6-terra declines from 0.685 to 0.377. xMemory declines from 0.620 to 0.435. EverMemOS also improves on L2 from 0.137 on M1 to 0.284 on M2 and 0.253 on M3. Together, these results highlight the potential of hierarchical external memory organisation to preserve and aggregate evidence across multiple memory anchors.

Memory recall does not guarantee effective assistance. The relative strengths of retrieval and external memory systems reverse from L1 to L2. On L1, EverMemOS surpasses Full Context with GPT-5.6-terra in mean met (0.621 vs. 0.570), while EverMemOS (0.621) and xMemory (0.546) outperform all RAG systems (0.504–0.539). On L2, however, every RAG system (0.334–0.370) outperforms every external memory system (0.139–0.269). This advantage is only relative because all RAG systems remain below Full Context with GPT-5.6-terra (0.431), while BM25 and dense retrieval outperform GraphRAG and RAPTOR. Thus, strong recall by external memory systems does not translate into effective assistance, while RAG provides stronger but still limited support.

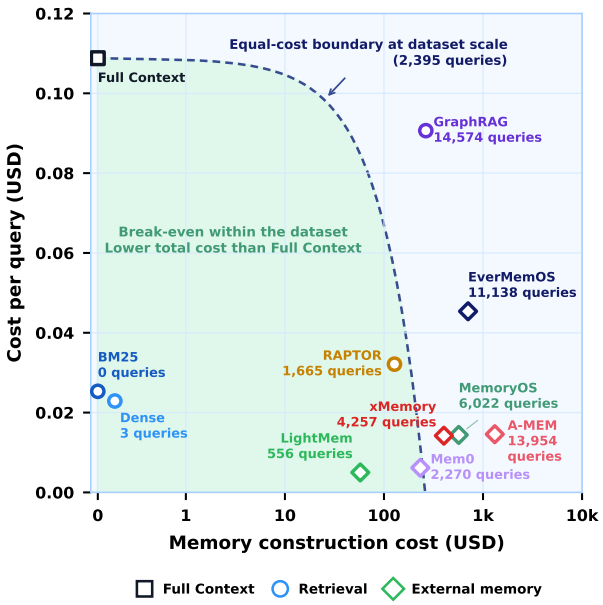


Figure 3: **Deployment cost efficiency.** The shaded region marks systems that amortize construction within all 2,395 queries relative to Full Context. Labels give the required query count. Judge costs are excluded.

End-to-end cost efficiency. To examine whether reduced context usage during queries offsets memory construction, we compute total deployment cost over the full dataset as construction cost paid once plus retrieval and generation costs across all queries. Figure 3 separates these components. At the observed workload of all queries, BM25, dense retrieval, LightMem, RAPTOR, and Mem0 cost less than Full Context with the same GPT-5.6-terra task agent. For example, dense retrieval costs $55.0 in total, compared with $260.4 for Full Context, while RAPTOR reaches $204.5 after amortising its $127.6 construction cost. Lower cost, however, does not imply stronger performance. LightMem costs only $69.6 but obtains the lowest L1 and L2 averages, whereas EverMemOS achieves the highest L1 average among external memory systems but costs $814.9 and requires 11,138 queries to break even. Overall, an efficient memory system must balance construction overhead and savings with effective memory recall and assistance. Detailed statistics for each stage are reported in Appendix F.1.

5 Analysis

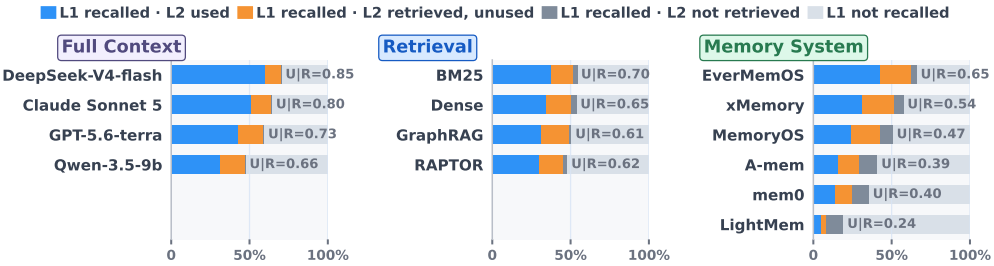


Figure 4: **Fate at L2 of memories recalled at L1.** Each bar covers 497 paired memories. Light gray marks those not recalled at L1. The other segments show whether recalled memories were used, retrieved but unused, or not retrieved at L2. $U \mid R$ is the blue share among recalled memories, $P(\text{used at L2} \mid \text{recalled at L1})$, not the bar length.

Recalled but not used. The gap between L1 and L2 does not show whether systems use memories they can recall, since the two levels contain different instances. We therefore identify 497 memories queried directly in L1 and required by an L2 request in the same group, and test whether each system uses in L2 what it recalls in L1 (Figure 4). Across all six memory systems, recalled but unused memories account for 21–27% of all pairs, versus 11–16% for models given full context. Even EverMemOS matches their recall (66% vs. 47–70%) but uses fewer recalled memories (65% vs. 73–85%). This fraction correlates more strongly with the L2 ranking in Table 3 ($\rho = 0.89$) than recall itself does ($\rho = 0.56$). Why are memories recalled at L1 left unused at L2? We split unused memories into those absent from the model input and those present but unapplied. (i) For memory systems, 36% of unused memories never entered the L2 input. This retrieval failure was cue dependent. L1 questions usually identify the target memory, whereas L2 requests often do not. Four of the six systems retrieved 84–100% of the same memories when L1 named them directly, and

input coverage across all pairs rose from 62% to 83% as lexical overlap increased. (ii) The other 64% were in the input and still not used in the reply. This failure depends on the cue as well, with memories already in the input used more often when requests shared their wording or were framed around the same matter. Without such cues, memory systems can thus fail twice, in retrieving the relevant memory and in applying it when present. These failures motivate retrieving evidence independently of request wording and binding available memories to the response. Analyses by generation model and request type appear in Appendix F.2.

Finding 1 Explicit relational evidence organisation recovers memories that the request does not name.

Better generation cannot recover evidence omitted by retrieval. We introduce **Relational Evidence Refinement (RER)**, which makes relations among retrieved memories explicit and follows them to supplement missing supporting evidence without relying on the wording of the request. We evaluate RER on xMemory and EverMemOS because of their leading L1 performance, and on flat LightMem to test generality. Missing evidence accounts for 74% of unused memories in LightMem, compared with 22% in xMemory and 14% in EverMemOS, as shown by the dark grey segments in Figure 4. Table 4 confirms this pattern. RER gives flat LightMem the largest average gains (+8.4 L1 met and +6.3 L2 $\bar{S}$), because it omits most evidence, and smaller gains to hierarchical xMemory and EverMemOS, whose evidence is more complete. On L1 M2/M3, anchor coverage exceeds met for xMemory (0.64–0.68 vs. 0.44–0.46) and EverMemOS (0.80–0.85 vs. 0.54–0.56). RER narrows this gap because integrating evidence across members and turns requires the connecting memories to be present.

Table 4: **Relational Evidence Refinement** by memory dependency; Avg weighted by instance counts, green = gain over baseline (pp).

Metric		EverMemOS		xMemory		LightMem	
		Baseline	+RER	Baseline	+RER	Baseline	+RER
L1 met	M1	0.679	0.693 (+1.45)	0.620	0.641 (+2.11)	0.290	0.380 (+8.98)
	M2	0.540	0.544 (+0.39)	0.455	0.494 (+3.86)	0.133	0.216 (+8.13)
	M3	0.561	0.566 (+0.50)	0.435	0.456 (+2.14)	0.178	0.235 (+5.71)
L2 $\bar{S}$	M1	0.137	0.160 (+2.28)	0.133	0.157 (+2.32)	0.192	0.215 (+2.32)
	M2	0.284	0.294 (+1.08)	0.223	0.245 (+2.22)	0.131	0.207 (+7.52)
	M3	0.253	0.294 (+4.04)	0.220	0.238 (+1.83)	0.094	0.161 (+6.71)
	Avg	0.247	0.265 (+1.77)	0.203	0.225 (+2.19)	0.139	0.202 (+6.28)

Finding 2 Effective assistance requires binding memories that are already present to the response.

The larger share of the loss comes from memories that reach the L2 input but remain unused. They account for 78% of unused memories for xMemory and 86% for EverMemOS. This application failure helps explain why EverMemOS falls from third on L1 to tenth on L2 and xMemory from fifth to thirteenth. We introduce **Assistance Constraint Binding (ACB)**, which determines each remembered fact's subject, scope, status, and applicability, then binds the validated conditions to generation. Table 5 again matches the decomposition: ACB improves xMemory and EverMemOS most on M1 (+6.8 and +6.6 $\bar{S}$), where the relevant facts are already available and the experiment isolates whether they are interpreted and enforced, while LightMem gains less because its primary bottleneck is missing evidence, which ACB cannot recover. When combined, the two modules perform best for every system on both levels, but their gain is smaller than the sum of their separate gains because each unused memory falls under only one bottleneck. Appendix F.2 provides the component analysis with confidence intervals and details of both methods.

Table 5: **Assistance Constraint Binding** on L2 by memory dependency.

L2 metric	EverMemOS		xMemory		LightMem	
	Baseline	+ACB	Baseline	+ACB	Baseline	+ACB
M1 $\bar{S}$	0.137	0.203 (+6.63)	0.133	0.201 (+6.77)	0.192	0.212 (+1.96)
M2 $\bar{S}$	0.284	0.306 (+2.17)	0.223	0.257 (+3.39)	0.131	0.159 (+2.80)
M3 $\bar{S}$	0.253	0.276 (+2.30)	0.220	0.243 (+2.17)	0.094	0.106 (+1.20)
Avg. $\bar{S}$	0.247	0.279 (+3.16)	0.203	0.242 (+3.94)	0.139	0.163 (+2.38)

6 Conclusion

We presented GroupAssistBench, a benchmark built from real-world group conversations with a deployed assistant. It evaluates memory recall and use over the same histories when the required memories are single, multiple, or conflicting. Required memories are anchored to source messages, and L2 rubrics translate real user feedback into criteria tailored to each request. Across LLMs given full context, retrieval methods, and memory systems, recall did not reliably translate into use. The strongest memory system surpassed the same backbone given full context on recall but lagged far behind on use, which declined further with multiple and conflicting memories. Relational Evidence Refinement and Assistance Constraint Binding improved all three tested memory systems by addressing complementary failures in retrieving relevant memories and applying those already available. We hope GroupAssistBench shifts memory evaluation from what systems recall to what they enable a group to do.

AI USE STATEMENT

In this work, we used generative AI tools to assist with synthetic data generation, dataset cleaning and reformatting, and the design and refinement of research methodology and experiments. We also used generative AI tools to create or modify scientific figures and to summarize and analyze existing literature. Generative AI tools were not used to formulate mathematical claims. All AI-assisted outputs were reviewed by the authors. For example, research ideas suggested by LLMs were checked against the existing literature through a manual literature review to guard against unoriginal or unsupported claims, and LLM-generated code was manually reviewed, tested, and validated for correctness. We take full responsibility for the final content of this work, including all text, claims, code, figures, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This study uses pseudonymized group-chat records. The use of these records for research was reviewed and approved through the data provider's privacy review process before the data were made available to the authors. Speaker identifiers are replaced with group-specific aliases, and message texts are rewritten to remove residual personal information, with safety-filtered content explicitly marked. The original source records remain restricted to the data provider and will not be included in any public release. Any release of derived benchmark artifacts will undergo an additional privacy and compliance review to further reduce disclosure and re-identification risks.

REPRODUCIBILITY STATEMENT

Upon acceptance, we will publicly release all author-owned code and privacy-reviewed derived benchmark artefacts necessary to reproduce the reported results. The benchmark release will include the rewritten and pseudonymized Chinese and English group histories, member profiles, queries, per-instance history cutoffs and split manifests, source-linked anchors and evidence spans, L1 reference and distractor atoms, and L2 rubrics with criterion types and weights. The code release will include the exact task and evaluator prompts, benchmark construction and validation pipeline, evaluator parsing and deterministic scoring code, result aggregation scripts, author-written baseline adapters, configurations, the relational evidence organisation and assistance constraint binding implementations, end-to-end run scripts, dependency specifications, and model and API version information. We will also release the model responses, evaluator judgments, and aggregate run metadata underlying the reported tables so that the reported scores can be verified without rerunning proprietary models. For third-party implementations or model weights that we cannot redistribute, we will provide versioned references together with our patches, adapters, and configurations. All released artifacts will be accompanied by explicit licenses or data-use terms. Original source records and the other restricted materials listed in the Ethics Statement will remain private.

REFERENCES

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. Healthbench: Evaluating large language models towards improved human health. *CoRR*, abs/2505.08775, 2025. doi: 10.48550/ARXIV.2505.08775. URL https://doi.org/10.48550/arXiv.2505.08775.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In Inês Lynce, Nello Murano, Mauro Vallati, Serena Villata, Federico Chesani, Michela Milano, Andrea Omicini, and Mehdi Dastani (eds.), *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Liang Ding. Adarubric: Task-adaptive rubrics for reliable LLM agent evaluation and reward learning. In *4th Workshop on Towards Knowledgeable Foundation Models at ACL 2026*, 2026. URL https://openreview.net/forum?id=zKMZysje4C.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *CoRR*, abs/2404.16130, 2024. doi: 10.48550/ARXIV.2404.16130. URL https://doi.org/10.48550/arXiv.2404.16130.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. Lightmem: Lightweight and efficient memory-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dyJ0GWpjJB.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. Benchmarking agent memory in interdependent multi-session agentic tasks. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=JHYmxqS9Jv.

Mengze Hong, Xia Zeng, Zeyang Lei, Sheng Wang, Chen Jason Zhang, Di Jiang, Taiming Fu, Jinfeng Huang, Mengqiao Liu, Qinghe Chang, Haosheng Zou, Qiongyi Zhou, Sijun He, Simonjmdeng, Haojing Huang, Zijian Li, Lucas Mu Li, Fubao Zhang, Mona Zhou, Wei Ma, Yuan Hua, Qi Zhu, Shuo Jiang, Chenxuan Ma, Yuanmeng Zhang, Jian Song, Minlong Peng, Di Liang, and Davey Chen. Uxbench: Benchmarking user experience in ai assistants, 2026. URL https://arxiv.org/abs/2606.09570.

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 Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 45836–45853, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2125. URL https://aclanthology.org/2026.acl-long.2125/.

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. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, pp. 9082–9090, New York, NY, USA, 2026b. Association for Computing Machinery. ISBN 9798400722592. doi: 10.1145/3770855.3817589. URL https://doi.org/10.1145/3770855.3817589.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *The Fourteenth International Conference on Learning Representations*, 2026c. URL https://openreview.net/forum?id=DT7JyQC3MR.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the age of AI agents. *CoRR*, abs/2512.13564, 2025. doi: 10.48550/ARXIV.2512.13564. URL https://doi.org/10.48550/arXiv.2512.13564.

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.

Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, Radha Poovendran, Gregory W. Wornell, Lyle H. Ungar, Dan Roth, Sihao Chen, and Camillo Jose Taylor. Personamem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory. *CoRR*, abs/2512.06688, 2025. doi: 10.48550/ARXIV.2512.06688. URL https://doi.org/10.48550/arXiv.2512.06688.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

Yifei Li, Weidong Guo, Lingling Zhang, Rongman Xu, Muye Huang, Hui Liu, Lijiao Xu, Yu Xu, and Jun Liu. Locomo-plus: Beyond-factual cognitive memory evaluation framework for LLM agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25085–25100, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1150. URL https://aclanthology.org/2026.acl-long.1150/.

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

Olukunle Owolabi. Socialmembench: Are AI memory systems ready for social group settings? *CoRR*, abs/2605.17789, 2026. doi: 10.48550/ARXIV.2605.17789. URL https://doi.org/10.48550/arXiv.2605.17789.

Zhe Ren, Yibo Yang, Yimeng Chen, Zijun Zhao, Benshuo Fu, Zhihao Shu, Bingjie Zhang, Yangyang Xu, Dandan Guo, and Shuicheng Yan. Gatemem: Benchmarking memory governance in multi-principal shared-memory agents. *CoRR*, abs/2606.18829, 2026. doi: 10.48550/ARXIV.2606.18829. URL https://doi.org/10.48550/arXiv.2606.18829.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GN921JHCRw.

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. URL https://openreview.net/forum?id=pZiyCaVuti.

Jianfei Xiao, Xiang Yu, Chengbing Wang, Wuqiang Zheng, Xinyu Lin, Kaining Liu, Hongxun Ding, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. AlpsBench: An LLM personalization benchmark for real-dialogue memorization and preference alignment. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '26, New York, NY, USA, 2026. Association for Computing Machinery. doi: 10.1145/3805712.3808634. URL https://doi.org/10.1145/3805712.3808634.

Wenya Xie, Shengming Zhou, Zelin Li, Pouya Parsa, Shuang Zhou, Xinheng Ding, Chinmay Arvind, Guanchu Wang, Vladimir Braverman, Ali Payani, Yantao Zheng, and Zirui Liu. Dynamicmem: A long-horizon memory benchmark in real-world settings. *CoRR*, abs/2606.22877, 2026. doi: 10.48550/ARXIV.2606.22877. URL https://doi.org/10.48550/arXiv.2606.22877.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

Jingbo Yang, Kwei-Herng Lai, Xiaowen Wang, Shiyu Chang, Yaar Harari, and Evgeniy Gabrilovich. Groupmembench: Benchmarking LLM agent memory in multi-party conversations. *CoRR*, abs/2605.14498, 2026. doi: 10.48550/ARXIV.2605.14498. URL https://doi.org/10.48550/arXiv.2605.14498.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, Yuandong Tian, and Jishen Zhao. AMA-bench: Evaluating long-horizon memory for agentic applications. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=GfQI92bpC1.

APPENDIX

A Construction Details 15
A.1 Data Source and Processing 15
A.2 Locating Requests 15
A.3 Anchoring Memories 15
A.4 Assigning Dependency Types 15
A.5 Building Instances 16
A.6 Quality Control 16
B Rubric Details 16
C Human Validation 17
D Dataset Statistics 19
E Implementation Setup 19
E.1 History Cutoffs 19
E.2 Retrieval and Memory Configuration 19
E.3 Generation and Output Validation 20
F Experiments 20
F.1 Cost and Token Efficiency 20
F.2 Intervention Analysis 20
G Case Studies 23
H Prompting and Evaluation Protocol 26
H.1 Task and Evaluation Protocol 26
H.2 Answer Generation Prompts 27
H.3 Evaluation Prompts 27
I Cost Accounting 28

A CONSTRUCTION DETAILS

This appendix expands the construction pipeline in Section 3.2 and follows the same order as Figure 2. Unless otherwise stated, the counts refer to the full validated pool over all groups, from which the evaluation set in Table 2 is sampled. All model calls in the pipeline use GPT-5.5, and their cost is reported in Appendix I.

A.1 DATA SOURCE AND PROCESSING

Selection. From the group conversations available to us we keep those with at least two members, sustained use of the assistant, and a complete message timeline, which yields 50 groups. Every group is used in full; no segment is removed for evaluation purposes.

Processing. All conversations are de-identified before they reach the authors. Member identifiers are replaced with per-group pseudonyms, message texts are rewritten to remove residual personal information while preserving the facts, values, and speaker of each message, and a small fraction of messages (2.5%) are replaced by placeholders of a safety filter. Non-text messages such as images and files are kept as placeholders that name the attachment. Each message keeps its speaker, timestamp, reply relation, and whether it is sent by a member or the assistant. The language of evaluation is described in Appendix H.1.

A.2 LOCATING REQUESTS

A *service point* is a message that mentions the assistant or immediately precedes an assistant reply. A lightweight classifier discards service points that are not requests, such as small talk or comments on a previous reply. Each remaining request is answered twice by the same model, once with the preceding history and once without it, and a comparator labels the contribution of the history as *strong*, *weak*, or *none*. Requests labelled *none* are discarded. Information that only appears in earlier assistant replies is excluded when judging the contribution, since it is generated content rather than memory of the group. Of 13,782 service points, 10,191 are requests and 7,926 are retained (6,772 strong and 1,154 weak).

A.3 ANCHORING MEMORIES

Anchors are proposed in one call with two passes. The first pass extracts candidate facts that the response depends on, and the second pass removes candidates whose absence would not change the response. Each anchor must state its fact without pointer expressions such as "the time mentioned above", and anchors that contain such expressions are rejected by a word list. The evidence span of each anchor is matched verbatim against the cited message, and anchors that fail are rejected (154 of 22,843 proposals). When a fact appears repeatedly, a rescan of the history anchors it to the earliest message that establishes it, and earlier lexical matches are flagged since a lexical match does not necessarily establish the same fact. Anchors whose source message is an assistant message are marked, and the reviewer in Appendix A.6 checks whether they relay a fact established by a member. Verified anchors are tracked in a per-group inventory that records which instances consume each anchor, which is used to select uncovered anchors for authored instances.

A.4 ASSIGNING DEPENDENCY TYPES

M1 and M2. Anchors from the same message and near-duplicate facts are merged into one *memory unit*, and anchors that only carry the profile of a member do not count. An instance with one unit is M1, and an instance with two or more units is M2.

M3. Candidate conflict pairs are detected from update expressions and from changes of the value in the same slot, and each new value is paired with the anchor that established the earlier value. Disagreements between members are included only if the conversation later resolves them. Each candidate pair receives three independent votes on the relation between the two values, including *true update*, *different slot*, *range extension*, *rephrase*, and *goal versus status*. Pairs with three *true update* votes are accepted, and the other pairs are not used unless a human annotator confirms them. Of 595 candidate pairs, 338 are accepted. For each accepted pair, we check that the new value is still current at the cutoff. If the matter changes again later, the cutoff is moved before the next change, or the pair is dropped. A M3 instance further requires at least two anchors on the same matter from members other than the one who makes the update, so that it keeps the structure of M2. The superseded value is excluded from the grounding criteria and enters the penalty criteria instead (Appendix B).

A.5 Building Instances

Native instances keep the original request of a member and place the cutoff at the request message. For authored instances, the query type is assigned by the pipeline in rotation from a list of nine types: single-fact recall, multi-entity attribute mapping, attribution, state tracking, constraint compliance, update resolution, aggregation, personalized assistance, and arbitration among members. Every factual premise in an authored query must cite a source message and a matching span, which is verified in the same way as an anchor. The cutoff of an authored instance is placed between $\delta_{\min} = 20$ and $\delta_{\max} = 200$ messages after its last anchor (Section 3.2), so that the instances cover a range of distances between a memory and its use, and no update of the same matter may fall within this range. How the cutoff restricts the input of each system at evaluation time is described in Appendix E.1. For L1 instances, the reference answer is split into atomic facts, one for each anchor, and for M3 instances the superseded value is recorded as a distractor. Rubrics of L2 instances are assembled as described in Section 3.3 and Appendix B.

A.6 Quality Control

Deterministic gates. An instance is rejected if (i) any anchor or cited premise does not precede the cutoff, (ii) any evidence span is not found verbatim in the cited message, (iii) the query reveals a required memory, names an evaluation dimension, or, for M3 instances, contains words that hint at a change, (iv) the cutoff leaves almost no history, or (v) the rubric violates the constraints in Appendix B.

Language-model review. A reviewer model then checks (i) whether each anchor is necessary, by testing whether a smaller set of anchors is sufficient, (ii) whether the query can be answered from its own wording, (iii) whether an anchor in an assistant message relays a fact established by a member or content generated by the assistant, and (iv) whether the two values of a M3 instance belong to the same matter. For every negative judgment the reviewer must quote the span that supports it, and quotes that are not found in the conversation invalidate the judgment. Instances that fail either layer are excluded from the pool but kept in the audit log with the reason. In total, 9,542 instances pass both layers and 440 are excluded.

B Rubric Details

This appendix complements the rubric composition (Equation 1) and scoring (Equation 3) in Sections 3.2 and 3.3.

Weights and constraints. Table 6 lists the weights. A rubric contains three to six positive criteria, at most three penalty criteria, and at most eight criteria in total. When more than four criteria are mandatory, those with the lowest weights are downgraded to optional criteria with weight 1 rather than removed.

Table 6: Weights of rubric criteria.

Kind	Criterion	Weight
Grounding	Core anchor of the instance	$+3$
Grounding	Other anchors	$+2$
Service	Mandatory requirement of the request	$+2$
Service	Optional improvement	$+1$
Penalty	See Table 7	-1 to -3

Penalty criteria. Table 7 lists the five failure patterns summarised from negative user feedback, together with the condition under which each pattern is instantiated for an instance and the number of feedback items behind it. Every instantiated criterion cites the message that provides its evidence. A penalty criterion is phrased as a description of the failure and is judged as met when the response commits the failure, as instructed in the L2 evaluation prompt (Table 18).

Derivation of the patterns. The five patterns come from negative feedback of our seed data, obtained in three steps.

1. *Collection.* Every member message that immediately follows an assistant reply and expresses a negative reaction (keyword filter): 423 candidates from five groups.
2. *Filtering.* A language model discards candidates that do not address the assistant, do not correct or object to the reply, or dispute subject knowledge rather than anything about the members: 97 remain (22.9%).

3. *Clustering.* The 97 corrections are grouped by what the assistant got wrong: a member's current state (23), a recalled fact (24), an explicit agreement or persona (21), a superseded value (12), an earlier statement or settled plan (17).

Four in five corrections concern the fidelity of the assistant's memory of members' states and facts rather than tone or knowledge, and the weights in Table 7 follow this ranking; the superseded-value weight is raised to -2 for M3 instances, where that failure is the defining risk. The 97 items are recorded with the member's wording, e.g., "not three, I did four sheets" (recalled fact) and "it should be geography now, not biology" (superseded value), and are released with the benchmark. The feedback is used once, to fix the set of patterns and their weights. Individual instances receive penalty criteria through the triggers of Table 7, which scan the anchors and the memory state of the instance; feedback from the instance's own group is not consulted.

Table 7: Failure patterns of penalty criteria and their triggers.

Failure pattern	Trigger	Feedback	Weight
Distorting a recalled fact	An anchor contains a numeric or other exact value that is not already covered by a grounding criterion	24	-2
Contradicting a settled plan	The group has settled a plan that is relevant to the request before the cutoff	17	-1
Violating an explicit agreement or member constraint	A constraint is stated explicitly in the group, rather than inherited from a profile, and is relevant to the request	21	-2
Misreading the current state of a member	A member states a current state within 150 messages before the cutoff	23	-3
Relying on a superseded value	The instance contains a verified conflict pair	12	-2

C Human Validation

We validated three things by hand: the instances themselves, the rubrics of the L2 instances, and the criterion-level judgments of the automatic judge. All three studies use the same sample of 120 instances drawn from the evaluation set, stratified by task level, memory dependency, and source (10 per cell), excluding instances that the pipeline had marked as superseded, and the same three annotators: one author and two annotators from the data provider, all fluent in Chinese and familiar with the group-chat setting. Annotators worked independently and did not see the outputs of the gates, the model reviewer, or the judge. An earlier annotation of the same sample by a second author was excluded: a joint review found that it had applied a stricter reading of the guidelines than the guidelines intend (it rejected 24% of the instances, against 5–10% for the three annotators, and agreed with them on only 71–79% of the instances), and the guidelines were clarified before the annotation reported here. We report raw agreement rather than a chance-corrected coefficient: valid judgments dominate every annotator's marginals, so coefficients such as Cohen's κ have almost no room above chance and are uninformative here.

C.1 Instance validity

The annotators judged each instance on four questions: (Q1) whether each anchor is established by the cited message, (Q2) whether each anchor is necessary for a good response, (Q3) whether the reference facts (L1) or the rubric (L2) are valid, and (Q4) for M3 instances, whether the two values concern the same matter and the new value is current at the cutoff. Each judgment is binary; an instance is *valid* if it passes all applicable questions. Table 8 reports the results.

By majority vote, 95.8% of the instances are valid (93.3% L1, 98.3% L2; 95.0/92.5/100% for M1/M2/M3), and every question is accepted by at least 97.5% by majority. Agreement is uniform across the three pairs (90–93% on the instance level, 93–98% per question), with no annotator systematically stricter than the others: of the 480 judgments, 464 are unanimous, 15 are rejected by a single annotator, and one is rejected by two. The unanimous rate (84.2%) is lower than the majority rate because an instance must pass up to four independent checks. The five instances rejected by majority fail on an anchor whose stated fact over-interprets a tentative message (Q1, 2 instances) or on an anchor that is not needed for the response together with a reference answer or rubric that goes beyond the request (Q2 and Q3, 3 instances). We did not verify anchor completeness, i.e., whether a required memory is missing, because it requires reading the whole history before the cutoff; completeness relies on the two-pass extraction and the counterfactual check of Appendix A. The annotation cards and all judgments are released with the benchmark.

Table 8: Human validation on 120 evaluation instances by three annotators. *Majority*: at least two of three accept; *unanimous*: all three accept; agreement is the pairwise agreement for the three annotator pairs. Q4 applies to the 40 M3 instances only. An instance is valid when every applicable question passes.

		Valid, per annotator (%)			Valid (%)		
Question	n	1	2	3	Majority	Unanimous	Agreement (%)
Q1 Anchor established by cited message	120	99.2	97.5	97.5	98.3	96.7	98.3 / 98.3 / 98.3
Q2 Anchor necessary	120	96.7	96.7	95.8	97.5	94.2	98.3 / 96.7 / 96.7
Q3 Answer key or rubric valid	120	97.5	95.8	95.0	97.5	93.3	98.3 / 96.7 / 95.8
Q4 Conflict pair valid (M3)	40	97.5	95.0	97.5	100.0	90.0	92.5 / 95.0 / 92.5
Instance valid (all questions)	120	95.0	91.7	90.0	95.8	84.2	93.3 / 91.7 / 90.0

C.2 Rubric quality

The instance-level check asks whether a rubric is valid as a whole. To examine the rubrics more closely, the annotators judged the 60 L2 instances of the sample on three further questions: (R1) whether every required service criterion is a reasonable, checkable requirement for the request rather than a restatement of it or a generic demand; (R2) whether every penalty criterion cites a fact relevant to the request and can be decided from the response; and (R3) whether a response that satisfies the rubric would be a good response in the situation, with no missing requirement, no contradiction, and no clearly misplaced weight. Majority-vote acceptance is 98.3% for R1, 95.0% for R2, and 96.7% for R3; 90.0% of the rubrics pass all three by majority and 86.7% unanimously. Pairwise agreement is 0.93–1.00 per question and 0.90–0.97 on all three. Penalty criteria are the weakest component: the two rubrics rejected unanimously fail on R2 or R3, and M3 rubrics have the lowest pass rate (80.0%, against 90.0% and 100% for M1 and M2), consistent with the fallback rule that derives a penalty from the superseded value when no user feedback is available (Appendix B).

C.3 Agreement between the judge and human annotators

We then checked the criterion-level judgments of the automatic judge. For each of the 60 L2 instances, the annotators independently marked every criterion as met or not met for three real responses (full-context GPT-5.6-terra, dense retrieval, and EverMemOS), presented in random order under anonymous labels and without access to the judge's output, giving 384 criteria $\times$ 3 responses $=$ 1,152 judgments per annotator. Table 9 compares the judge with the majority vote of the annotators and with the annotators among themselves.

Table 9: Criterion-level agreement between the automatic judge (GPT-5.6-sol) and three human annotators on 60 L2 instances $\times$ 3 responses. Human–human is the mean pairwise agreement; judge–human compares the judge with the annotators' majority vote. Precision and recall are for the *met* label with the majority vote as reference. Miss: the majority marks a criterion met and the judge does not; extra: the reverse. For penalty criteria, *met* means the response committed the error.

		Agreement		Judge vs. majority		Disagreement (%)	
Criterion type	n	Human–human	Judge–human	Precision	Recall	Miss	Extra
Grounding (G)	351	0.85	0.86	1.00	0.67	14.2	0.0
Service, required (S)	300	0.88	0.89	0.99	0.80	10.7	0.3
Service, bonus (S)	180	0.86	0.86	0.92	0.70	11.7	2.2
Penalty (P)	321	0.99	0.95	0.48	1.00	0.0	4.7
All	1,152	0.90	0.89	0.94	0.74	8.9	1.7

The judge agrees with the human majority on 89.3% of the judgments, the same level as the annotators agree with one another (mean pairwise 0.898, range 0.88–0.93), and it agrees with the strictest annotator (0.915) more often than that annotator agrees with the other two. Its errors are one-directional. On grounding and service criteria the judge only misses: it never credits a grounding criterion the majority rejects, but fails to credit 14.2% of the grounding criteria and about 11% of the service criteria that the majority accepts, typically when the response states the fact in paraphrase rather than in the wording of the rubric. On penalty criteria it only over-penalizes: it never overlooks an error the majority found, but flags 4.7% of the penalty criteria that the majority considers not violated. Both directions lower the scores; neither raises them. At the instance level, recomputing score and pass from the human judgments with the

formula of Section 3.3 gives a pass agreement of 88.9% and a score correlation of $r = 0.84$ ($\rho = 0.84$) with the judge, above the mean between annotators (86.3% and $r = 0.81$); the judge's mean score is 0.13 below the human majority, and the ranking of the three systems is identical under the judge and under every annotator. Responses that decline to answer (25 of the 60 EverMemOS responses) are not a source of disagreement: on these the judge and the majority agree on 95–100% of the criteria. We therefore read the automatic scores as a conservative estimate that preserves the ordering of systems, and in Section 5 we treat the share of "present but unused" memories as an upper bound that the miss rate of the judge would reduce by roughly one seventh.

D Dataset Statistics

GroupAssistBench comprises 2,395 questions grounded in conversations from 50 groups: 1,374 questions for memory recovery (L1) and 1,021 for memory use (L2). Each question is assigned to one of three memory dependency levels. M1 (*Single*) requires one substantive memory, M2 (*Integrative*) combines complementary memories, and M3 (*Evolving*) resolves updates, conflicts, or differences in applicability. The corpus contains 80,299 message records and 4,924,259 tokens. Table 10 summarizes the question distribution and variation across groups. Group histories range from 2,887 to 972,839 tokens, with ten groups exceeding 100,000 tokens and four exceeding 200,000 tokens.

Table 10: Statistics of GroupAssistBench across 50 groups. The upper block shows question counts by evaluation level and memory dependency. The lower block reports corpus totals and the minimum, mean, and maximum per group.

Question distribution				
Level	M1	M2	M3	Total
L1	776	448	150	1,374
L2	221	650	150	1,021
Total	997	1,098	300	2,395
Corpus scale				
Statistic	Total	Min.	Mean	Max.
Questions	2,395	4	47.90	110
Message records	80,299	68	1,605.98	4,442
History tokens	4,924,259	2,887	98,485.18	972,839

History lengths are measured over the complete packaged conversation of each group using `o200k_base` from `tiktoken` 0.12.0. Records are serialized as `#turn [speaker] content` and joined by newlines. The packaged text retains at most 2,000 characters per message. These counts precede the question cutoff and model context budget, and exclude member profiles, evaluation queries, and system prompts. They describe corpus length, not the context tokens supplied to a model for an individual question. Message counts refer to stored records, including distinct records that share a turn index.

E Implementation Setup

E.1 History Cutoffs

Every question, whether taken from the conversation or authored for the benchmark, is assigned a history cutoff c_i to prevent answer leakage from later messages. Only conversation turns with index $t < c_i$ are eligible for context or memory construction. The turn at c_i and all subsequent turns are excluded from the historical input. The evaluation query is supplied separately. The same temporal restriction applies to both native and authored questions.

E.2 Retrieval and Memory Configuration

We build on the official implementations where available and adapt their data interfaces and configurations to the group conversations in GroupAssistBench. The adaptations handle conversation records, speaker identifiers, and question cutoffs. Table 11 summarizes how each method accesses the available history.

Table 11: Overview of history access and memory retrieval in the evaluated systems.

Method	History and memory access
Full Context	Supplies conversation history before the cutoff directly to the task agent, with context fitting when needed.
BM25	Uses overlapping windows of 10 messages with a stride of 5 messages. Ranks these windows and eligible profile entries with BM25 and retrieves the top 20 units.
Dense retrieval	Uses the same overlapping windows of 10 messages with a stride of 5 messages and eligible profile entries. Retrieves the top 20 units using embedding similarity.
GraphRAG	Uses local search over entities and relations to retrieve graph evidence and associated source passages.
RAPTOR	Builds a hierarchy of messages and summaries and retrieves relevant representations across its levels.
Mem0	Extracts and updates memory entries from the conversation, then retrieves entries relevant to the query.
LightMem	Compresses incoming messages, organizes them by topic, and retrieves relevant memories.
A-MEM	Organizes linked memory notes and retrieves relevant notes together with their neighbours.
MemoryOS	Combines recent messages, retrieved sessions, and stored knowledge to provide query context.
xMemory	Uses adaptive retrieval over themes, semantic memories, and episodes.
EverMemOS	Retrieves episode memories by combining embedding and BM25 rankings through reciprocal rank fusion.

E.3 Generation and Output Validation

The API compatibility configuration for GPT-5.6-terra and GPT-5.6-sol uses the supported default temperature of 1, either explicitly or by omitting the temperature parameter. This accommodates the model interface rather than imposing a temperature shared by all model families. Generation uses the adapters' output limits and retry procedures.

The evaluator returns structured JSON. Its parser checks that judgments cover all required atoms or rubric criteria. An incomplete judgment is treated as a generation failure rather than filling missing judgments with false values. The final comparison additionally requires a nonempty answer, a matching source cutoff, and a judgment from the designated evaluator. Section H presents the answer generation and evaluation prompts.

F Experiments

F.1 Cost and Token Efficiency

We separate memory construction from retrieval and answer generation. Each evaluation item contributes one query, and costs are aggregated across the full dataset. In Figure 3, construction cost is amortised over additional queries, while the mean cost of each query remains fixed.

Table 12 reports token consumption for construction, retrieval, answer generation, and judging across the runs used in the main comparison. Construction reports LLM tokens and embedding tokens separately, while retrieval reports their combined total. GPT-5.6-sol serves as the judge model. Judge tokens are included for reproducibility but excluded from the deployment curves because judging is not performed during deployment.

For the deployment comparison, total cost after Q queries is the construction cost plus the mean cost of retrieval and generation for each query multiplied by Q. Table 13 reports the values underlying Figure 3. GraphRAG combines its retrieval LLM allocation with one query embedding, while MemoryOS, xMemory, and EverMemOS use their recorded query embedding usage.

The accounting reveals two distinct efficiency regimes. BM25, dense retrieval, LightMem, and RAPTOR recover their construction costs within the dataset, but their low costs do not yield the strongest task scores. Conversely, several structured memory systems improve L1 recovery while requiring more queries to recover their construction costs. Overall memory efficiency should therefore consider evidence utility and service quality together with construction costs and expenditure during queries, rather than equating shorter context with a better memory system.

F.2 Intervention Analysis

Re-slicing by generation model. All retrieval and memory systems generate with GPT-5.6-terra, so the full-context run of the same model is the matched comparison. Among the 497 paired memories, the fraction of memories that are in the model's input and are then used is 66% for full context with GPT-5.6-terra, 61–62% for BM25, Dense and EverMemOS, 50% for xMemory, 43–46% for A-MEM, MemoryOS and Mem0, and 24% for LightMem. The gap between EverMemOS and full context in Figure 4 is thus partly a generation-model gap (Claude Sonnet 5 and

Table 12: Token consumption by stage. All numeric entries are in millions of tokens (M). Construction reports LLM and embedding tokens separately, while retrieval combines both types.

Method	Build LLM	Build embed.	Retrieval	Answer	Judge	Total
Full Context (Claude Sonnet 5)	0.0	0.0	0.0	249.4	3.3	252.8
Full Context (Terra)	0.0	0.0	0.0	183.4	3.2	186.6
Full Context (DeepSeek)	0.0	0.0	0.0	156.0	3.4	159.4
Full Context (Qwen)	0.0	0.0	0.0	164.6	3.7	168.2
BM25	0.0	0.0	0.0	37.8	3.2	41.0
Dense retrieval	0.0	9.7	0.2	33.8	3.2	46.8
GraphRAG	41.9	13.1	69.2	65.8	3.2	193.1
RAPTOR	22.1	17.6	0.2	50.9	3.2	94.0
Mem0	94.8	11.9	0.0	6.1	2.9	115.7
LightMem	13.9	0.0	3.6	4.5	2.6	24.6
A-MEM	260.0	28.6	13.5	20.1	3.3	325.5
MemoryOS	121.6	10.0	0.2	19.4	3.6	154.7
xMemory	116.1	11.4	0.3	21.0	3.3	152.0
EverMemOS	272.9	3.2	0.2	73.9	3.4	353.5

Table 13: Deployment cost statistics relative to Full Context with GPT-5.6-terra. Construction cost is paid once; the cost of each query includes retrieval and answer generation but excludes judging. N^* is the number of queries required to offset construction cost, and C is the cumulative cost for all 2,395 queries in the dataset.

Method	Construction cost (USD)	Cost/query (USD)	N^*	C (USD)
Full Context (Terra)	0.0	0.1088	—	260.4
BM25	0.0	0.0253	0	60.6
Dense retrieval	0.2	0.0229	3	55.0
GraphRAG	264.6	0.0906	14,574	481.5
RAPTOR	127.6	0.0321	1,665	204.5
Mem0	232.9	0.0062	2,270	247.7
LightMem	57.7	0.0050	556	69.6
A-MEM	1,314.7	0.0146	13,954	1,349.6
MemoryOS	568.5	0.0144	6,022	602.9
xMemory	402.6	0.0142	4,257	436.6
EverMemOS	706.3	0.0454	11,138	814.9

DeepSeek-V4-flash use 72% and 79% of in-input memories); the remaining memory systems fall behind the matched full-context model even when the memory is present.

Re-slicing by request type: native versus authored. L2 requests are either written by group members (native) or authored by the pipeline; authored requests are checked not to reveal the memory, but they ask about the matter the memory bears on, whereas native requests are ordinary messages. The two types have almost the same distribution of lexical overlap with the memory (30%/49%/21% vs. 35%/48%/17% of memories at zero, low and higher overlap), and memories enter the input equally often (69% vs. 67% for memory systems). Yet memories in the input are used less often on native requests: 58% vs. 71% for full context, 38% vs. 61% for retrieval, and 30% vs. 51% for memory systems, and the gap persists within each overlap band (e.g., at zero overlap 56% vs. 65%, 32% vs. 52%, 19% vs. 41%). The cue that drives memory use is therefore not shared wording but whether the request is framed around the matter at all; questions in existing memory benchmarks are, and requests in real group chats are not.

Relational Evidence Refinement. RER operates between retrieval and answer generation. Given a query and its retrieved evidence, it identifies items that concern the same members, events, or decisions, organises them to expose complementarity, chronology, updates, and conflicts, and adds missing supporting items from the eligible candidate pool when necessary. It preserves the original history cutoff and source identifiers.

Table 14: Performance gains from Relational Evidence Refinement (RER), Assistance Constraint Binding (ACB), and their combination. Values are percentage point gains over the corresponding baseline; brackets show the 95% intervals used in Figure 5.

System	Level	RER	ACB	RER + ACB	Best condition
xMemory	L1 met	2.74 [1.45, 4.01]	4.22 [2.66, 5.77]	4.74 [3.08, 6.40]	RER + ACB
	L2 $\bar{S}$	2.19 [1.18, 3.21]	3.94 [2.70, 5.21]	4.64 [3.36, 5.92]	RER + ACB
EverMemOS	L1 met	1.00 [−0.09, 2.11]	0.27 [−0.98, 1.49]	1.64 [0.38, 2.91]	RER + ACB
	L2 $\bar{S}$	1.77 [0.74, 2.85]	3.16 [1.77, 4.55]	4.43 [3.02, 5.82]	RER + ACB
LightMem	L1 met	8.35 [6.60, 10.18]	0.44 [−0.65, 1.53]	9.18 [7.46, 10.95]	RER + ACB
	L2 $\bar{S}$	6.28 [4.92, 7.67]	2.38 [1.48, 3.34]	7.03 [5.71, 8.39]	RER + ACB

Table 15: Anchor coverage under Baseline, RER, ACB, and RER + ACB for hierarchical external memory systems. Values report the fraction of required source turns retained in the evidence.

System	Level	Condition	M1	M2	M3
xMemory	L1	Baseline	0.760	0.641	0.677
		RER	0.760	0.641	0.677
		ACB	0.760	0.639	0.674
		RER + ACB	0.762	0.640	0.682
	L2	Baseline	0.482	0.444	0.515
		RER	0.482	0.444	0.515
		ACB	0.477	0.445	0.515
		RER + ACB	0.480	0.445	0.513
EverMemOS	L1	Baseline	0.858	0.796	0.849
		RER	0.831	0.752	0.817
		ACB	0.861	0.795	0.849
		RER + ACB	0.830	0.750	0.815
	L2	Baseline	0.586	0.645	0.681
		RER	0.464	0.523	0.581
		ACB	0.588	0.649	0.679
		RER + ACB	0.456	0.516	0.576

Assistance Constraint Binding. ACB interprets retrieved memories as constraints on the current assistance response. For each constraint, it records the affected subject, scope, content, status, applicability, source identifiers, and an exact supporting quotation. Code verifies the source and quotation, removes duplicate or inapplicable constraints, and supplies the validated constraint block as a separate input to answer generation. Thus, ACB changes how retrieved evidence governs the response without adding new memories.

Table 14 provides the complete numerical results underlying Figure 5. RER changes the organisation or completeness of retrieved evidence, ACB binds retrieved evidence to constraints on the assistance response, and their combination applies both interventions. L1 changes are measured in Atom Recall points, while L2 changes are measured in rubric score points.

Table 15 reports anchor coverage separately because it describes the evidence supplied to the task agent rather than answer quality. The table includes all four experimental conditions for the two hierarchical systems, whose memory units retain links to source turns. LightMem is omitted because its rewritten units combine and redistribute content from multiple turns, preventing unique coverage measurement at the turn level.

The component analysis exposes different system bottlenecks. RER contributes most to flat LightMem through evidence completion, whereas ACB contributes more to hierarchical xMemory and EverMemOS on L2 through response conditioning. Their combination performs best for every system on both levels, although the gains overlap rather than add directly. Figure 5 shows the same pattern with confidence intervals; the two modules thus address the two bottlenecks of Figure 4 rather than the same one.

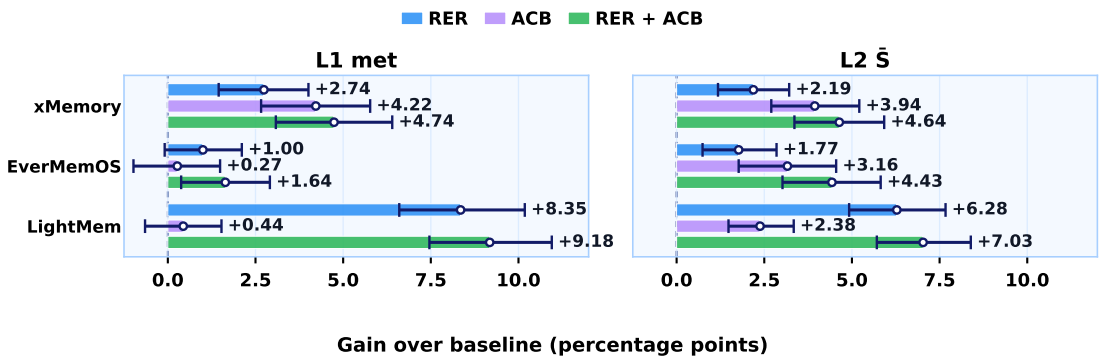


Figure 5: **Component analysis of RER and ACB.** Bars show gains over baseline in percentage points, with 95% confidence intervals.

G Case Studies

The following examples cover all six cells of the task taxonomy and are selected from the 2,395-item common evaluation set. Questions, memories, and responses are translated and paraphrased, and identifying details are replaced while preserving the evaluated relations. All responses use GPT-5.6-terra as the task agent and GPT-5.6-sol as the judge. For L1, the displayed score is Atom Recall; for L2, it is the normalized rubric score. In L2 rubrics, G denotes grounding, S denotes service quality, and P denotes a penalty.

L1M1 (Single Memory): Identifying a loading failure

Question: A mobile error trace was shared earlier. What failed, and at which loading step?
Memory anchor:

A_1. An Android application crashed with an `UnsatisfiedLinkError` while loading `libjiagu.so`.

Reference answer: The application crashed because loading `libjiagu.so` raised an `UnsatisfiedLinkError`.

Correct response (EverMemOS): "The Android application crashed with an `UnsatisfiedLinkError` while loading `libjiagu.so`."
Judge result: Atom Recall $= 1.00$ $(1/1$ reference atom recovered).

Incorrect response (LightMem): The response attributes the trace to a different software issue.
Judge result: Atom Recall $= 0.00$ $(0/1)$; the failed library-loading step is not recovered.

L1M2 (Multi-Memory Integration): Combining members' reactions

Question: Three members reacted to the video just shared. Match each member to what they said, and list the three reactions.
Memory anchors:

A_1. Member A said the video left them stunned.
A_2. Member B found the sound painfully harsh.
A_3. Member C said they were also frightened.

Reference answer: Member A was stunned by the video; Member B found its sound painfully harsh; Member C was also frightened.

Correct response (xMemory): "A was stunned, B found the sound harsh, and C was also frightened."
Judge result: Atom Recall $= 1.00$ $(3/3$ reference atoms recovered with the correct speakers).

Incorrect response (LightMem): The response assigns the reactions to different members and adds unrelated details.
Judge result: Atom Recall $= 0.00$ $(0/3$ reference atoms recovered).

L1M3 (Memory Updates and Conflicts): Resolving an updated itinerary

Question: What is the current plan for tomorrow morning and noon, and what happened to the location that required a reservation?
Memory anchors:

A_1. The group will attend a banquet at noon.
A_2. Tickets for the originally planned location could not be reserved.
A_3. The morning plan was updated to visit an old-town area instead.

Reference answer: Visit the old town in the morning, attend the banquet at noon, and note that the original location could not be booked. The updated morning plan supersedes the earlier itinerary.

Correct response (EverMemOS): "Visit the old town in the morning and attend the banquet at noon; the original location could not be booked."
Judge result: Atom Recall $= 1.00$ $(3/3$ current reference atoms recovered).

Incorrect response (LightMem): "Follow the earlier sightseeing route and decide the remaining stops later."
Judge result: Atom Recall $= 0.00$ $(0/3)$; the response returns unrelated itinerary fragments and misses the current facts.

L2M1 (Single Memory): Drafting a group announcement

Request: Turn the group's purpose and participation expectations into a concise announcement.
Memory anchor:

A_1. The group is for casual discussion of astrological charts, signs, and transits.

Rubric:

G, +3. State the remembered purpose of the group.
S1, +2. Use a concise, itemized announcement format.
S2, +2. Require friendly discussion, inclusion of newcomers, and tolerance of disagreement.
S3, +2. Encourage a rational attitude and prohibit fear-inducing or absolute claims.
P1, -2. Penalize any distortion of the remembered group purpose.

The positive-score denominator is 9.

Correct response (LightMem): "Group notice: discuss charts, signs, and transits casually; welcome newcomers, respect disagreement, and avoid alarmist or absolute claims."
Rubric judge result: G and S1–S3 are met; P1 is not triggered. Score: $(3 + 2 + 2 + 2)/9 = 1.00$.

Incorrect response (EverMemOS): "Please keep the group active and enjoy chatting with one another."
Rubric judge result: G and S1–S3 are missed; P1 is not triggered. Score: $0/9 = 0.00$.

L2M2 (Multi-Memory Integration): Coordinating field attendance

Request: The discussion mixed who planned to attend an event, the transport arrangement, and who would remain behind. Summarize the current personnel arrangement in two parts.

Memory anchors:

A_1. Member A explicitly said they would attend the event on site.

A_2. The coordinator confirmed that five on-site members would travel together in one car.

A_3. Member B explicitly said they would remain with the rear-support team rather than attend on site.

Rubric:

G1, +3. State that Member B remains with rear support and is not in the on-site group.

S1, +1. Organize the answer into on-site attendance and rear support.

G2, +2. State that the five on-site members travel together in one car.

S2, +2. Preserve the single-car arrangement without inventing another roster or convoy.

G3, +2. Include Member A among those attending on site.

P1, -3. Penalize misattribution of another member's external-partnership status.

P2, -2. Penalize violation of the group's progress-reporting agreement.

P3, -1. Penalize contradiction of an existing contingency plan.

The positive-score denominator is 10.

Correct response (EverMemOS): "On site: Member A joins the other attendees, and the five-person group travels together in one car. Rear support: Member B stays behind and does not join the on-site team."
Rubric judge result: G1–G3 and S1–S2 are met; P1–P3 are not triggered. Score: $10/10 = 1.00$.

Incorrect response (LightMem): "Member A will attend on site. Member B can join the field team separately, and another vehicle can be arranged if needed."
Rubric judge result: Only G3 is met; G1–G2 and S1–S2 are missed; P1–P3 are not triggered. Score: $2/10 = 0.20$.

L2M3 (Memory Updates and Conflicts): Applying an updated production plan

Request: Explain how an anniversary-video plan changed over time, identify who proposed or confirmed each change, and write a production brief that follows the final decision.

Memory anchors:

A_1. The initial proposal was a five-minute complete retrospective.
A_2. A member requested a maximum duration of three minutes.
A_3. Another member argued for a three-minute highlight edit to protect completion rate.
A_4. The final decision confirmed a three-minute montage synchronized to highlight moments.

Rubric:

G1, +3. Recover both the final three-minute duration and highlight synchronization.
S1, +1. Present the complete chronological update chain.
G2, +2. Attribute the duration limit and completion-rate rationale to the correct members.
S2, +2. Explain why the complete retrospective was replaced by a shorter montage.
G3, +2. Recover the initial five-minute proposal.
P1, -2. Penalize treating the superseded five-minute proposal as current.
P2, -2. Penalize violating the group's existing rule against public-shaming acknowledgments.
P3, -1. Penalize contradicting the agreed announcement time.

The positive-score denominator is 10.

Correct response (EverMemOS): "The proposal moved from a five-minute retrospective to a three-minute maximum, then to a three-minute highlight montage to protect completion rate. Produce the final three-minute edit with cuts synchronized to the selected highlights."
Rubric judge result: G1–G3 and S1–S2 are met; P1–P3 are not triggered. Score: $10/10 = 1.00$.

Incorrect response (BM25): "The team discussed an anniversary video, production pressure, delays, and a later editing handoff."
Rubric judge result: G1–G3 and S1–S2 are missed; P1–P3 are not triggered. Score: $0/10 = 0.00$.

H Prompting and Evaluation Protocol

This section documents the task and evaluation protocol used for the reported experiments. We evaluate each method end to end using its official implementation where available, together with its method-specific memory construction and retrieval procedures. All retrieval and external memory systems use GPT-5.6-terra as the task agent, and all reported outputs are scored by GPT-5.6-sol with the same task-level criteria and scalar scoring rules. Method-specific context wrappers and response formats are retained where required by the corresponding implementations; these differences are treated as part of each complete system rather than as changes to benchmark questions or scoring.

H.1 Task and Evaluation Protocol

The source group conversations, runtime instructions, and evaluation prompts are in Chinese to preserve the original meaning and conversational context. We also provide a complete English translation of the benchmark for reference and future research. The boxes below give an English schematic specification of the shared task and evaluation requirements; they are not verbatim transcriptions of every adapter's wrapper. Braced placeholders denote inputs inserted at runtime, and character limits refer to Chinese characters.

Across methods, we hold fixed the benchmark query, history cutoff, language, task definition, and scoring criteria. The supplied context differs according to the history or memories made available by each method, while adapter-specific wrappers preserve the corresponding implementation interface. L1 outputs are scored against the same reference atoms and L2 outputs against the same instance-specific rubrics. Auxiliary diagnostics, where present, are excluded from the reported metrics.

H.2 Answer Generation Prompts

Table 16 summarizes the common answer requirements shared across implementations. Adapter-specific wrappers serialize full histories, retrieved passages, or stored memories according to the corresponding method interface, and the L1 response format is selected by task level.

Table 16: Common Answer Requirements

Instructions

You are an AI assistant in a group chat. Answer the question using the supplied context.

1. Use only information supported by the context. Preserve the exact meaning of names, numbers, dates, and other factual details.
2. When records contain `#index [speaker] content`, larger indices denote later turns. Refer to members using their supplied speaker identifiers.
3. If there are updates or conflicting claims, determine which information applies to the question, considering explicit corrections, decision authority, and the requested time or scope. Do not treat a superseded value as current.
4. If evidence is insufficient, explicitly state what cannot be determined from the context. Do not invent missing information or guess unsupported facts.
5. Answer in Chinese. Address the requested points concisely without repeating the question.

Response format

For L1, output only the following two fields:
`ANSWER`: Give the requested conclusions, with at most 20 Chinese characters per point. Separate multiple points with enumeration commas.
`WHY`: Briefly state the basis for the answer in at most 25 Chinese characters. Do not repeat `ANSWER`.

For L2, answer the service request directly, following any format specified in the question and grounding the response in the supplied context.

Input

Task level: {`L1 or L2`}
Context: {`context`}
Question: {`query`}

H.3 Evaluation Prompts

Tables 17 and 18 summarize the common core of the L1 and L2 evaluation instructions. Reference atoms and rubric criteria are supplied only to the evaluator, not to the task agent. Every reported output is judged by GPT-5.6-sol under the same atom or rubric criteria, and auxiliary diagnostics do not enter the reported scores.

Table 17: L1 Evaluation Prompt

You are a strict evaluator. Given a question, a list of atomic facts in the reference answer, and a candidate answer, independently determine whether the candidate recovers each atom.

1. Each atom is either correct or incorrect. Allow paraphrases, but require equivalent numbers, names, dates, and times, such as 42.5 and 42.50, or 22:30 and 10:30 p.m.
2. If the answer is vague about an atom, asks a question instead, or says it does not know, mark that atom false.
3. Extra content outside the reference does not affect atom judgments. However, if the answer gives a value contradicting a reference atom, mark that atom false.
4. Separately count `extra`: concrete factual points that the reference did not request at all. Count each independent number, person, date, or factual matter. Do not count introductory wording, repetitions of the question, or explanatory filler. This count does not affect the atom judgments.

Question: {`query`}
Atoms: {`numbered reference atoms`}
Answer: {`candidate answer`}

Output only a JSON object with `atoms`, a Boolean array containing exactly {`number of atoms`} entries, `extra`, an integer, and `reason`, at most 25 Chinese characters.

For L2, each numbered criterion includes its kind, signed weight, and check text. The penalty marker is rendered as `Penalty` in the English specification.

Table 18: L2 Evaluation Prompt

You are a strict evaluator. Independently judge whether the answer meets each criterion.

1. `[G]` denotes memory grounding. The answer must explicitly state the specific fact referred to by the criterion, such as a number, name, date, chapter, or key point from an original statement. Appropriate behaviour without that specific fact is false.
2. `[q]` denotes a quality criterion. Judge according to its wording.
3. `[Penalty]` denotes a failure criterion. `met=true` means the answer actually commits the described error. A negatively phrased criterion such as "must not misremember" or "must not contradict" is a prohibition. For such a criterion, true still means violating the prohibition, not complying with it. When uncertain, return false.

Question: {`query`}
Answer: {`candidate answer`}
Criteria: {`numbered criteria with kind and signed weight`}

Output only a JSON array with exactly {`number of criteria`} entries. Each entry contains `n`, the criterion number, and `met`, a Boolean judgment.

The evaluation code computes the scalar scores from the returned judgments rather than asking the evaluator to assign a final score directly. A penalty marked true contributes its negative weight in Equation 3.

I Cost Accounting

We account for both computational and human costs throughout benchmark development. Computational costs include repeated data construction and validation, agent-assisted engineering, baseline adaptation and reruns, and final generation and evaluation. Recorded usage is priced according to the corresponding model, while incompletely logged adaptation and rerun costs are conservatively estimated from the main baseline runs. Human labor is grouped into benchmark design and protocol development, dataset construction and annotation, and quality assurance, adjudication, and experimental validation.

The human-labor subtotal reports compensation actually paid to contributors. It covers benchmark and protocol design; construction and review of seed cases, queries, anchors, ground truths, and rubrics; M3 update and conflict annotation; L1–L2 pair verification; quality review and adjudication; and baseline adaptation, reruns, result verification, and statistical analysis. Converting paid workdays to eight-hour days, benchmark design involved two experts for approximately 15 days each (240 person-hours), dataset construction involved four experts for approximately 15 days each (480 person-hours), and quality assurance and validation involved ten experts for approximately two days each (160 person-hours). Automated model execution is excluded from person-hours; only active construction, review, debugging, monitoring, and validation time is counted.

Table 19: Computational and human costs for benchmark development and baseline evaluation. We price benchmark construction, iteration, and agent-assisted engineering at GPT-5.5 rates ($5 per million input tokens and $30 per million output tokens). Baseline construction, generation, and judging use their recorded model prices, including GPT-5.6-sol for evaluation. Human labor costs reflect compensation actually paid to contributors. Paid workdays are converted to person-hours using an eight-hour workday, and person-hours, effective hourly rates, and USD values are rounded for presentation.

Computational costs

Component	Input tokens	Output tokens	Cost (USD)
Benchmark construction	1,701.3M	62.1M	10,370
Preliminary benchmark iteration	315.8M	8.4M	1,832
Agent-assisted benchmark engineering	2,583.4M	11.1M	13,249
M-axis, L-axis, and combined analysis experiments	706.5M	31.2M	1,871
Baseline adaptation and reruns (estimated)	961.1M	103.9M	3,076
Main baseline evaluation	1,922.3M	207.8M	6,151
Computational subtotal	**8,190.4M**	**424.6M**	**36,549**

Human labor

Component	Person-hours	Rate (USD/hour)	Cost (USD)
Benchmark design and protocol development	240	20	4,800
Dataset construction and human annotation	480	20	9,600
Quality assurance, adjudication, and experimental validation	160	15	2,400
Human labor subtotal	**880**	—	**16,800**
Total project cost	—	—	**53,349**