

MEMORY AS A SHARED SPACE: RETRIEVAL INTERFACES AND VIEW-FILTERED INDEXES FOR PERSISTENT AGENTS

Yanming Guo

The University of Sydney
AIDC

yguo0337@uni.sydney.edu.au

Haixin Wang

AIDC

nix@ai-dc.ai

YanJun Lu

AIDC

lucas@ai-dc.ai

ABSTRACT

A widely used open-source agent runtime keeps each agent’s conversation history in a local session store and lets the agent recall it through a lexical search tool; the stores grow without bound and other agents cannot read them. We treat memory as a persistent region of the space through which agents communicate: committed turns are embedded once into one shared index, lexical statistics are kept per view, and each agent retrieves through a view — its own records and those other agents made public — that the index evaluates. Holding the reader model and the memory content fixed, we compare the runtime’s session search (default and compact output), a shell, a compact retrieval tool with lexical, dense or hybrid scoring, prefetching and full context on LongMemEval, and six ways of sharing on LongMemEval-Org, a split of LongMemEval across public and private agents. The hybrid tool used $0.51\times$ the tokens of the default session search at a similar number of model calls ($0.98\times$); accuracy was 1.7 points lower (95% CI -5.6 to $+1.7$), so non-inferiority was not established, and truncating the runtime’s own output saved as many tokens but cost 6.7 points. Dense scoring did not beat BM25; with ten times longer histories, hybrid scoring was 9.5 points below BM25 (exploratory, $p = 0.125$). A shared view answered 91.7% of questions whose evidence other agents held, similar to asking those agents (90.0%) at $0.20\times$ the tokens and $0.21\times$ the time, and none of 8,019 retrieved units came from a private agent; a policy prompt without the view kept answers private here but let 1,523 private units into the context. Of six pre-registered primary hypotheses, two held after Holm adjustment (one by construction), the token saving held only before adjustment, and three did not hold. All results come from one reader model, one small English encoder and an English chat benchmark.

1 INTRODUCTION

A persistent agent answers people for months, and what it can do depends on what it can recall. Agent runtimes keep each agent’s past conversations in a local session store and let the agent recall them with tools: Hermes Agent, for example, writes every message to an SQLite database with two full-text indexes and gives the agent a lexical session-search tool, and an agent with a shell can also query the file itself (Nous Research, 2026). These stores grow without bound. On the shared runtime pool of the deployment that motivated this study, 295 agents hold session stores totalling 38.5 GB; the median store is 0.47 MB and the largest 8.5 GB. Upstream reports describe a 9.4 GB store whose index maintenance saturated a disk, and compaction summaries that came back through search as 73k-character results (issues 68858 and 43175).

Two costs follow. The first is the *cost of recall*: each search is a model round trip, the agent guesses keywords and reads windows of earlier messages, and every result stays in the context of the remaining calls. The second is *isolation*: a store belongs to one agent, so what the user told the finance agent is invisible to the sales agent, and the only bridge is a message from one agent to another, which costs generated tokens on both sides and exposes whatever the answering model chooses to write.

Research on agent memory has concentrated on what to store and how to organise it — hierarchical paging (Packer et al., 2023), extracted facts (Chhikara et al., 2025), temporal knowledge graphs (Ras-

[mussen et al., 2025](#)), linked notes ([Xu et al., 2025](#)) — evaluated by answer accuracy on long-term memory benchmarks ([Wu et al., 2025](#)). Three questions that decide how a deployment should store memory remain open. How much of the cost of recall comes from the *interface*, the way an agent reaches its memory, when content and model are fixed? Can an organisation-wide index let agents use each other’s memory without exposing what must stay private, and at what cost compared with asking? And which properties of a vector index matter once every query carries a per-agent filter? Proposals to add vector or hybrid search to the runtime’s session search are open upstream without measurements ([issue 44075](#)); filtered vector search has been studied with generic attributes, not with the views of an agent organisation ([Patel et al., 2024](#); [Gollapudi et al., 2023](#)).

This paper treats memory as a region of the space through which agents communicate. Our earlier work moved agent state out of processes into records ([Guo, 2026a](#)) and replaced messages between agents by a shared task space read through views ([Guo, 2026b](#)); that space lives as long as a task. We add a persistent, organisation-wide *memory space*: the committed turns of all agents, embedded once into one index, which each agent reads through a view that admits its own units and units other agents made public. Holding the agent loop, the model and the memory content fixed, we compare the ways an agent can reach its memory — the runtime’s own session search (default and compact output), a shell, a compact retrieval tool with lexical, dense or hybrid scoring, prefetching and full context — and the ways agents can share: isolated stores, fan-out over stores, one index with views, one index without views (with and without a privacy instruction), and asking another agent.

Our contributions are:

- a minimal formulation of agent memory as a persistent region with views, extending the access closures of [Guo \(2026b\)](#), used to state four testable observations: view safety (with a side channel through lexical collection statistics), truncation under post-filtering, a decomposition of the token cost of recall, and the cost of fan-out (Section 3);
- a controlled comparison of memory-access interfaces inside one agent loop on LongMemEval, with paired measurements of accuracy, calls, tokens and time, a decomposition of tokens into prefix, results and output, and a scale comparison from 115k-token to 1.5M-token histories (Sections 5.1–5.2);
- LongMemEval-Org, a multi-agent split of LongMemEval with public and private agents, and a comparison of sharing mechanisms with a unit-level audit of private exposure (Section 5.3);
- a same-host measurement of index backends — Redis 8 query engine and vector sets, PostgreSQL with pgvector, SQLite FTS5 — under views of different selectivity, with footprint, fan-out and write-path delay (Section 5.4).

In short: a compact retrieval tool halves the tokens of recall without fewer model calls, and the saving depends on how the tool is designed, not on vectors (Section 5.1); dense scoring helped neither at 115k nor at 1.5M tokens of history (Section 5.2); one index with views gives a team of agents each other’s public memory at a fifth of the cost of asking, and keeps private units out of every context, which an instruction alone does not (Section 5.3); at the scale measured, the backend matters through how it applies the view, not through its speed (Section 5.4).

2 RELATED WORK

Agent memory and long-term memory benchmarks. Memory systems decide what is stored and how: MemGPT pages context between tiers ([Packer et al., 2023](#)), Mem0 extracts and consolidates facts ([Chhikara et al., 2025](#)), A-Mem links notes ([Xu et al., 2025](#)) and Zep builds temporal graphs ([Rasmussen et al., 2025](#)). Runtimes such as Hermes keep raw sessions in a per-agent SQLite store searched lexically ([Nous Research, 2026](#); [SQLite Consortium, 2026](#)); in our harness agents keep no process state, only records ([Guo, 2026a](#)). Benchmarks test recall over long conversations ([Maharana et al., 2024](#); [Wu et al., 2025](#)); on LongMemEval, GPT-4o reading the full history reaches 60.6% against 87.0% with only the evidence sessions (self-reported). Memory layers report fewer tokens or calls than their baselines (self-reported; e.g. [Fang et al., 2026](#)), but each changes what is stored and how it is read at once. We fix the stored rounds, reader and content, and vary only the interface.

Retrieval for agents: lexical, dense, agentic and long-context. BM25 (Robertson & Zaragoza, 2009) remains a strong zero-shot baseline (Thakur et al., 2021); dense retrievers beat it on open-domain question answering (Karpukhin et al., 2020), and on LongMemEval’s own round-level retrieval (self-reported Recall@5: BM25 0.472, Contriever 0.589, Stella 0.660). Fusing rankings by reciprocal rank (Cormack et al., 2009) is sensitive to its parameters (Bruch et al., 2024). Retrieval is interleaved with reasoning for multi-step questions (Trivedi et al., 2023); agent performance depends on the tool interface (Yang et al., 2024), and file and grep tools are a strong baseline for agents that read their own records. Long-context models can beat retrieval on average at higher cost (Li et al., 2024) and use long inputs unevenly (Liu et al., 2024). In our semantic-graph study, raw tools reached the accuracy ceiling and tokens followed what stayed in the conversation (Guo, 2026c).

Vector search and filtered nearest-neighbour search. Graph indexes such as HNSW (Malkov & Yashunin, 2020) give high recall at low latency, compared by recall and speed on generic datasets (Aumüller et al., 2020). Vector database systems treat queries over attributes and vectors together as a core difficulty (Wei et al., 2020; Pan et al., 2024); filtered search builds label-aware graphs (Gollapudi et al., 2023), partitions (Gupta et al., 2024) or predicate subgraphs (Patel et al., 2024). Products document their behaviour: pgvector filters after the index scan and added iterative scans in 0.8.0 (Kane, 2026); the Redis query engine chooses between brute-force and batched filtered search (Redis, 2026a); vector sets cap the filtering effort (Redis, 2026b). We measure the recall an agent obtains when its view selects one owner among many, against exact search under the same view.

Shared memory in multi-agent systems, access control and privacy. Blackboard architectures organise problem solving around a shared workspace (Nii, 1986), whereas many LLM frameworks coordinate through conversation (Wu et al., 2024). Team memories are organised hierarchically (Zhang et al., 2025); Collaborative Memory adds private and shared tiers under time-varying policies (Rezazadeh et al., 2025); lineage gating blocks results derived from data the requester may not see (Sangaraju & Vissa, 2026); scope-bound declassification ties disclosure to recipient and purpose (Xu et al., 2026). CoMemBench reports a sharing–isolation trade-off (Zhao et al., 2026), and in a web setting asking content agents lagged centralised retrieval (Zhong et al., 2026). Retrieved stores can leak private records (Zeng et al., 2024) or accept injected ones (Dong et al., 2025). Scoped memories in these systems differ in where the scope is enforced (application, prompt or store) and in whether derived data inherit it; ours is the simplest case — unit labels enforced in the index — and we measure what it prevents against no view and against an instruction, and what it costs against messages, extending the access closures of Guo (2026b) to memory.

3 METHOD

3.1 PROBLEM FORMULATION: A MEMORY REGION WITH VIEWS

Prior objects. In the agent-independent harness of Guo (2026a) an agent keeps no process state: an execution of agent A_i reads its record $d_{i,\ell,t}$ and commits a new one, and an *access closure* $\text{Acc}(i, \ell, t)$ bounds what it may read. Guo (2026b) give a task a *space* $S_\tau = \{R_1, \dots, R_M\} \cup \{H\}$ of regions plus a hand-off region H ; an execution of A_i observes only its *view* $V_i(S_\tau) = \{R \in S_\tau : R \in \text{Acc}(i)\}$. The space lives as long as the task. Across tasks, what an agent has learned survives only in its own *session store* Σ_i (in Hermes Agent, an SQLite file with full-text indexes; Nous Research, 2026), which only A_i can read, through tools.

Memory units and the memory region. A *memory unit* is a tuple $m = (o, v, t, s, x)$: the agent $o(m)$ whose committed turn produced it, a visibility $v(m) \in \{\text{private}, \text{public}\}$ set by the owner’s definition, the commit time $t(m)$, the session $s(m)$, and a text $x(m)$, here a chunk of one user–assistant round. The *memory region* at time t is

$$\mathcal{M}(t) = \bigcup_i \Sigma_i(t), \quad (1)$$

the union of all agents’ committed units. Unlike H , $\mathcal{M}$ is persistent and organisation-wide; unlike Σ_i , it is one object.

Memory views. The closure of Guo (2026b) applied to memory gives agent A_i the view

$$V_i^{\mathcal{M}}(t) = \{m \in \mathcal{M}(t) : o(m) = i\} \cup \{m \in \mathcal{M}(t) : v(m) = \text{public}\}. \quad (2)$$

Retrieval instead of mounting. A task view in Guo (2026b) is materialised as files; a memory view cannot be, because it grows without bound. The agent reads it through a *retrieval operator*

$$\rho_k(q; V) = \text{top-}k \sigma(q, m)_{m \in V}, \quad (3)$$

where σ is a lexical score (BM25; Robertson & Zaragoza, 2009), a dense score $\cos(\phi(q), \phi(x(m)))$ for an embedding model ϕ , or reciprocal rank fusion of the two (Cormack et al., 2009). Metadata predicates (a date range) restrict V further.

Write path. When A_i commits a turn, its units enter Σ_i at once and enter the index after a delay δ (chunking, embedding, insertion): $\{m : t(m) \leq t - \delta\} \subseteq \mathcal{M}(t)$. Section 5.4 measures δ .

3.2 OBSERVATIONS

Let E_i be the set of units of other agents whose content can reach A_i 's context through retrieval during a task (its *exposure*, as in Guo, 2026b). The four statements below are elementary; the experiments test each.

Proposition 1 (View safety). *If ρ_k evaluates the predicate of Eq. 2 inside the index, then $\rho_k(q; V_i^{\mathcal{M}}) \subseteq V_i^{\mathcal{M}}$ for every query q , so exposure through retrieval is limited to own and public units whatever the model generates. If lexical scores use collection statistics of $\mathcal{M}$ rather than of the view, the ranking itself depends on private units, because a private unit changes the document frequencies of the terms it shares with public units.*

The guarantee is label-level: content that a public agent restates, and memories derived from private units, are outside it. The second sentence is why our lexical indexes are built per view.

Proposition 2 (Post-filter truncation). *If an approximate index returns the K' nearest units ignoring the view and the predicate is applied afterwards, and a fraction s of units lies in the view independently of its distance to q , the number of returned in-view units is $\text{Binomial}(K', s)$; when $K's \leq k - 1$, fewer than k are returned with probability at least one half, since the median of a binomial lies within one of its mean.*

For one agent among $N = 180$ ($s \approx 0.006$) and $K' = 40$ the expected yield is 0.22. Relevance correlates with ownership in practice, so real yields are higher; pre-filtering, or an iterative scan that continues until k survivors are found, removes the truncation.

Proposition 3 (Cost of recall). *Let an agent answer with c rounds of tool calls whose results add $w_1, \dots, w_c$ tokens, generating $g_0, \dots, g_c$ tokens in its $c + 1$ model calls, on a prompt prefix of P tokens (system prompt, tool schemas, question). Since each call re-reads the prefix and everything appended before it, its input tokens are*

$$I = (c + 1)P + \sum_{l=1}^c (c - l + 1) (g_{l-1} + w_l), \quad (4)$$

of which, with a prefix cache, only $P + \sum_l (g_{l-1} + w_l)$ are uncached. Prefetching k units of length u costs one call with $P + ku$ input tokens.

Three levers follow: the number of rounds c , the result length w and the prefix P , which includes the tool's own schema. The speed of the index enters none of them.

Proposition 4 (Fan-out). *Searching N per-agent stores costs at least N store queries. If the agent issues them one round at a time, input tokens grow quadratically in N (Proposition 3); if it issues them in parallel within a round, linearly. Merging N BM25 lists does not give the BM25 ranking of the union, because each store has its own document frequencies.*

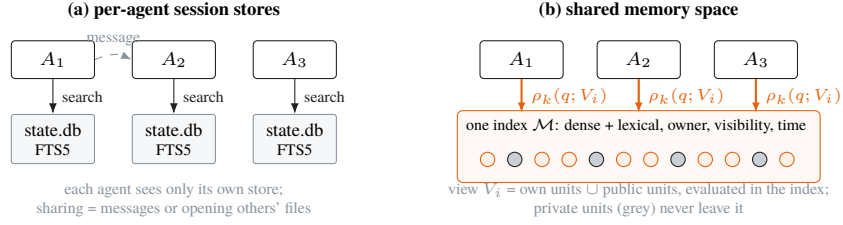


Figure 1: (a) Deployed runtimes keep one session store per agent; recall is a lexical search or a shell command against that file, and other agents’ knowledge is reachable only by messaging them. (b) The memory space: committed turns of all agents are indexed once; each agent retrieves through its view (Eq. 2), with the predicate evaluated inside the index.

3.3 ACCESS INTERFACES AND SYSTEM

Ways to reach memory. We hold the agent loop, the model and the memory content fixed and vary the interface (Figure 1): (i) *session search* over Σ_i — the runtime’s own lexical tool, which ranks messages with FTS5, groups them by session and returns the top session hydrated with a window of messages of up to 4,000 characters, plus scroll and read operations; we also run it with a *compact* output (the matching message and one neighbour on each side, at most 1,000 characters each, no bookends); (ii) a *shell* in a read-only sandbox containing Σ_i , where the agent writes its own `sqlite3` and `grep` commands; (iii) a *memory tool* that calls ρ_k on the view and returns k compact results (date, session, round, the matching chunk), plus a read operation for neighbouring rounds; (iv) *prefetch*, which places $\rho_k(\tau; V_i)$ for the question τ in the prompt and offers no tool. Interfaces (i) and (ii) are what deployed agents do today; (iii) and (iv) read the memory space. Between (i) and (iii) under BM25 several things change together — result length and format, unit (message or round chunk), grouping, tool schema and date filters — and the compact variant of (i) separates result length from the rest; BM25, dense and hybrid scoring within (iii) differ only in scoring; (iii) and (iv) differ in iteration, query formulation, k and the read operation.

Implementation. Units are user–assistant rounds split into chunks of at most 1,000 characters with 200 characters of overlap; continuation chunks repeat the start of the user turn. Dense vectors come from a 384-dimensional encoder (Xiao et al., 2024; BAAI, 2023) in fp32 with CLS pooling, computed once per chunk on a workstation GPU and at query time on the pool’s CPU (cosine similarity between the two paths: 1.000000 on 40 sampled chunks); no tokens, no network. The vector index is the query engine of Redis 8 (Redis, 2026a): an HNSW graph (M 16, $EF_{\text{construction}}$ 200, EF_{runtime} 10) with TAG and NUMERIC fields for owner and time, queried with the view as a filter. Each view also has its own SQLite FTS5 index (SQLite Consortium, 2026), with the tokenizer of the runtime’s session store, so that BM25 statistics never include units outside the view (Proposition 1); dense vectors are stored once, lexical entries once per view that contains the unit. Hybrid ranking fuses the top 50 of each list by RRF with constant 60. The harness of Guo (2026a) meters every model call.

4 EXPERIMENTAL SETUP

Research questions. **RQ1 (interface):** does retrieval from the memory space lower the cost of recall — model calls, tool calls, tokens, time — relative to agentic search over a session store, without lowering accuracy, and how does the difference change as history grows? **RQ2 (scope):** can one index with views let agents use each other’s memory at lower cost than per-agent fan-out or asking another agent? That no private unit is exposed is a design requirement; the experiment checks the implementation and contrasts it with no view and with a privacy instruction. **RQ3 (backends):** which properties of the index decide recall under views, latency and footprint? Hypotheses and decision rules were fixed in a dated design note before any model call (2026-10-09 03:30 AEDT, SHA-256 prefix 30fcd8314c685c62); Appendix A lists every pre-specified test with its outcome and every deviation from the note, and labels later analyses as exploratory.

Benchmarks. LongMemEval (Wu et al., 2025) gives each question its own chat history and a date; answers require recalling a fact the user or the assistant stated (ss-u, ss-a), a preference (ss-p), combining sessions (ms), reasoning about time (tr) or tracking an updated fact (ku); 30 questions are unanswerable and must be declined. We use the cleaned release. S histories have about 50 sessions (115k tokens), M histories about 500 (1.5M tokens). We draw 30 questions per type with a seeded shuffle (180 questions) for the main comparison, and for the scale comparison take the first 7 per type of that order with their M histories (42 questions, 283,653 chunks; Section 5.2).

LongMemEval-Org splits 120 non-abstention questions of the sample over an organisation of six agents. Agent a_0 receives the question; a_1 – a_3 make their memory public; a_4 , a_5 keep it private. Sessions are assigned uniformly at random, except the evidence sessions: in the *public* variant they go to public agents (two evidence sessions to two different agents), in the *private* variant (42 questions) to private agents, where the correct behaviour is to say the information is not available. The asking agent never holds evidence, so the isolated condition answers almost nothing by construction.

Conditions. All conditions use the same agent loop: DeepSeek V4.1 Flash (model alias deepseek-flash, accessed 2026-10-08/09; DeepSeek-AI, 2026) with low reasoning effort, OpenAI-compatible tool calling, at most ten tool rounds and then a forced answer, and one system prompt that differs only in the paragraph describing memory access (Appendix B). On LongMemEval: *none* (no memory), *full* (the whole history in the prompt; 60 questions), *session search* (the runtime’s own FTS5 tool, unmodified, on a store written by the runtime’s own storage code), *session search, compact*, *shell*, the memory tool with BM25, *dense* or *hybrid* scoring, and *prefetch* (eight hybrid results in the prompt, no tool). On LongMemEval-Org: *isolated* (memory tool over a_0 ’s own units), *fan-out* (the runtime’s session search, which accepts the name of another agent’s store, over a_0 and the three public agents), *shared* (memory tool over the view of Eq. 2), *no view* (the same index without the predicate), *no view + policy* (the same, with an instruction not to use or reveal what a_4 and a_5 hold; private variant only) and *ask agent* (memory tool over own units plus a tool that sends a question to a public agent, which answers with its own memory tool; its calls are counted). Memory tools in LongMemEval-Org use hybrid scoring.

Measurements. Accuracy is judged with the official per-type prompts of LongMemEval with DeepSeek V4.1 Flash as judge (the authors used GPT-4o; Appendix E). On private-evidence questions we judge whether the answer declines (correct) and whether it states the gold answer (a *leak*, as pre-specified), and we audit *exposure* by replaying every logged retrieval of each run against the same index and view and counting retrieved units owned by private agents. Per question we log model calls, tool calls, input tokens split by prefix-cache hit and miss, output tokens, list-price cost, wall time, retrieval time and whether a tool result contained an evidence session (a session-level upper bound on evidence recall). Conditions are compared on the same questions with paired bootstrap intervals (10,000 resamples; 100,000 for the primary tests) and exact McNemar tests, with Holm adjustment over the five primary hypotheses that have a test statistic (H2b is a count rule); runs of all conditions of a stage were interleaved in random order, with 6–12 concurrent agents.

Infrastructure and budget. Redis 8.10.2 with its query engine, PostgreSQL 17.11 with pgvector 0.8.7, the runtime’s session-store code (Hermes Agent 0.21.5), the agent driver and the metering proxy run in one resource envelope (3 GB memory and 6 cores for the agent runs; 3.7 GB with a 2 GB Redis limit for the backend study) on a shared runtime pool of a production deployment. Every model call was metered (Appendix G). Code, prompts, the LongMemEval-Org builder and logs: distributed-harness/bench6 (released with the paper).

5 RESULTS

5.1 RQ1: THE INTERFACE SETS THE COST OF RECALL

Table 1 answers RQ1 on S histories; Figure 3 in Appendix D plots accuracy against tokens.

Tokens halve; calls hold; accuracy and time move little. The hybrid memory tool used $0.51\times$ the tokens of the runtime’s session search (95% CI 0.39–0.69) and $0.58\times$ its list-price cost; the median per-question ratio was 0.57 (0.52–0.64), and the tool was cheaper on 79% of questions. Against the

Table 1: Memory-access interfaces on LongMemEval S (180 questions; one run each). Accuracy with Wilson 95% interval; calls, tokens and cost are means per question (cost at list price, fen = 0.01 yuan); time is the median wall time per question; *Evid.* is the share of questions for which some tool result contained an evidence session (a session-level upper bound on evidence recall). Full context is reported on its 60-question subset in the text.

Condition	Acc. (%)	95% CI	Calls	Tool calls	Tokens (k)	Uncached (k)	Cost (fen)	Time (s)	Evid. (%)
No memory	8.9	[5.5, 14.0]	1.00	0.00	0.7	0.2	0.50	1.4	0.0
Session search (FTS5)	93.9	[89.4, 96.6]	2.71	2.82	28.3	12.9	3.28	2.5	99.4
Session search, compact	87.2	[81.6, 91.3]	2.96	3.16	14.7	5.3	1.59	2.9	99.4
Shell grep/SQL	92.8	[88.0, 95.7]	4.79	6.04	24.0	6.9	2.49	5.7	97.2
Memory tool: BM25	90.6	[85.4, 94.0]	2.60	2.70	11.9	6.0	1.70	2.3	100.0
Memory tool: dense	89.4	[84.1, 93.1]	2.83	3.07	17.1	7.3	2.03	2.5	98.3
Memory tool: hybrid	92.2	[87.4, 95.3]	2.66	2.84	14.4	7.0	1.91	2.6	100.0
Prefetch (hybrid)	78.3	[71.8, 83.7]	1.00	0.00	2.4	2.0	0.69	1.3	99.4

pre-registered threshold of 0.7 the upper bound is 0.691 with 100,000 resamples (one-sided $p = 0.021$), but $p = 0.062$ after Holm adjustment: H1b holds at the unadjusted level only. Model calls did not fall ($0.98\times$, $0.92\text{--}1.04$; H1a not supported). Accuracy was 1.7 points lower (-5.6 to $+1.7$; McNemar $p = 0.549$), so non-inferiority at -3 points is not established (H1c). Mean wall time was $0.87\times$ ($0.78\text{--}1.00$); median times were 2.6 and 2.5 s.

Where the tokens go. Per question, session search started from a prefix of 1.4k tokens and appended 11.6k tokens of tool results; the hybrid tool 0.7k and 6.5k, with similar output and rounds. Two levers of Proposition 3 moved: w , because each session search hydrates the best session with messages of up to 4,000 characters (47.7k characters per question against 28.0k), and P , because the runtime’s tool description is longer and is re-read at every call. The saving is therefore specific to the runtime’s default output.

The saving depends on the tool’s design, not on vectors. Cutting the runtime’s own output to the matching message and its two neighbours saved as many tokens ($0.52\times$, $0.43\text{--}0.64$) but cost 6.7 points of accuracy (-11.7 to -2.2 ; $p = 0.012$), half of it on preference questions (66.7% against 86.7%). At the same tokens ($0.98\times$) the hybrid tool was 5.0 points more accurate than this compact search ($+0.0$ to $+10.0$; $p = 0.078$). Short results alone are not enough; the memory tool also differs in unit, ranking and read operation, which we did not separate. The scoring function mattered less: BM25 used $0.42\times$ the tokens of session search ($0.33\text{--}0.53$), fewer than hybrid scoring, at 3.3 points lower accuracy (-7.8 to $+1.1$); dense scoring alone used $1.43\times$ the tokens of BM25 and was the least accurate of the three (89.4%). Hybrid scoring was 1.7 points above BM25 (-1.7 to $+5.6$) and 2.8 above dense scoring (-1.7 to $+7.2$); H1d holds in direction only. The first search returned an evidence session for 98.3% of questions with BM25, 98.9% with hybrid and 95.6% with dense scoring.

Prefetch, shell and full context. Prefetch answered in one call with $0.08\times$ the tokens and $0.48\times$ the time of session search, but 13.9 points below the hybrid tool ($+8.3$ to $+19.4$; $p < 0.001$). Of the 25 questions it lost net, 8 were multi-session, 5 preference, 4 temporal, 4 knowledge-update and 4 assistant-fact questions (H1e supported for cost and direction; per-type samples of 30 are not tested). The shell matched session search in accuracy (92.8%; -1.1 points, -5.0 to $+2.8$) with $1.77\times$ its calls and $1.95\times$ its time; its first command reached evidence for 53.9% of questions, and its tokens were not the highest ($0.85\times$, $0.70\text{--}1.06$); its per-question token variation was the lowest of the agentic interfaces (coefficient of variation 1.12 against $1.30\text{--}1.95$) and its run-to-run difference the highest (Table 5), so H1g holds for calls and time only. Full context reached 90.0% on its 60 questions with 105.4k tokens per question; on the same questions the hybrid tool reached 96.7% with 14.0k.

5.2 RQ1 AT SCALE: FROM 115K-TOKEN TO 1.5M-TOKEN HISTORIES (EXPLORATORY)

We ran 42 questions on their M histories (19,920 sessions, 283,653 chunks; Figure 5 in Appendix D). The set was extended from an initial 18 questions after an unexpected drop of the hybrid tool, so we treat the comparison as exploratory; on this subset the S accuracies also differ from the main sample (hybrid 95.2% against 92.2%). With ten times more history every interface lost accuracy and every agentic interface used more tokens. BM25 lost 4.8 points and kept the highest accuracy at M (83.3%) with $1.94\times$ the tokens; session search and dense scoring lost 11.9 points each, prefetch 14.3 and

Table 2: LongMemEval-Org: sharing mechanisms. Accuracy, tokens, calls and median time on the 120 questions whose evidence public agents hold; on the 42 questions whose evidence private agents hold, the share of answers that decline (correct) and the number the judge found to state the private answer (leaks). *Exposed runs*: runs in which any retrieved unit was owned by a_4 or a_5 , from replaying every logged retrieval (public / private variant). *Fan-out opens only the stores of a_0 – a_3 .

Condition	Acc. (%)	Tokens (k)	Calls	Time (s)	Declined (%)	Leaks	Exposed runs (pub./priv.)
Isolated store	1.7	18.6	3.13	3.1	95.2	0/42	0/120/0/42
Fan-out session search	89.2	56.1	3.98	5.3	88.1	2/42	0*/0*
Ask another agent	90.0	75.4	12.66	13.8	85.7	2/42	0/120/0/42
Shared space (views)	91.7	15.0	2.60	2.4	88.1	3/42	0/120/0/42
Shared, no views	91.7	21.0	2.83	2.4	0.0	39/42	86/120/42/42
No view + policy prompt	–	–	–	–	90.5	2/42	–/42/42

hybrid scoring 21.4. At M the hybrid tool was 9.5 points below BM25: all 4 discordant questions favoured BM25 (exact McNemar $p = 0.125$), which is suggestive, not significant, and we do not test the difference between the S-to-M drops. Retrieval still reached the evidence: some tool result contained an evidence session for 40 of 42 questions with the hybrid tool and 41 with BM25. The failures we inspected were multi-session counts and updated facts in which units from other sessions about similar events entered the answer. Tokens of session search grew $1.48\times$ and those of the hybrid tool $1.30\times$, a ratio of 1.14, below the pre-registered 1.2 (H1f not supported). Prefetch stayed flat in tokens, model calls grew 1.08 – $1.13\times$ in the agentic conditions, and full context was not run at M.

5.3 RQ2: SHARING THROUGH ONE INDEX WITH VIEWS

Table 2 answers RQ2; Figure 6 in Appendix D plots it.

At similar accuracy, a view costs a fifth of asking. The isolated condition answered 1.7% by construction, so H2a (+90.0 points; Holm-adjusted $p < 0.001$) holds trivially; the informative comparisons are at similar accuracy. With the shared view the agent answered 91.7% of the public-evidence questions; the hybrid tool over one store holding the whole history answered 94.2% of the same questions, so splitting the history over six agents cost little. Fan-out — the runtime’s session search called once per store of a_0 – a_3 — reached 89.2% with 8.08 tool calls in 3.98 model calls per question; the agent issued about two searches per round, so its tokens grow linearly, not quadratically, in the number of stores (Proposition 4). The shared view used $0.27\times$ its tokens (0.19–0.37) and $0.48\times$ its time; a factor of about two of this gap is the interface difference of Section 5.1. Asking a colleague reached 90.0%, but every question started up to three more agent loops: 12.66 model calls, 75.4k tokens and a median of 13.8 s per question. The shared view was 1.7 points above it (–2.5 to +5.8) at $0.20\times$ the tokens (0.14–0.27) and $0.21\times$ the time (H2c supported; Holm-adjusted $p = 0.003$). Asking returns a generated answer instead of records; on temporal and multi-session questions the shared view answered 85.0% and 95.0% against 80.0% and 85.0%, on knowledge updates both 100.0%.

A predicate in the index keeps private units out of every context; without it, nothing in our setup did. We audited exposure unit by unit, replaying every logged retrieval against the same index and view. Under the view none of the 8,019 units retrieved in 162 runs was owned by a_4 or a_5 ; this follows from Proposition 1 and checks the implementation. Without the view, private units entered the context in 86 of 120 public-variant runs and in all 42 private-variant runs (933 units); the agent never declined and stated the private answer in 39 of 42. An instruction not to use or reveal what a_4 and a_5 hold changed the answers but not the context: the agent declined in 90.5% of questions and stated the private answer in 2, while 1,523 private units were in its context across all 42 runs, where another question, a weaker instruction or injected text could bring them out. By the pre-registered answer-level measure the view is not leak-free: the judge found the private answer in 3 shared-view, 2 fan-out and 2 ask-agent answers although no private unit was retrieved. H2b therefore holds for exposure (0 of 42 runs against 42 of 42 without the view) but not as pre-registered. The seven flags come from three questions (Appendix C): the gold answer of a knowledge-update question also occurs in a session not marked as evidence, a preference rubric is met by preferences the user stated in other sessions, and in a quotation question the model said it had no record and then quoted the text from general knowledge. The two flags of the instruction condition are on the first two of these questions.

5.4 RQ3: WHAT THE INDEX BACKEND CHANGES

Figure 2 answers RQ3 on the 122,942 chunks of the 180 S histories, each history standing for one agent; the 180 questions are the queries, and the ground truth is exact search under the same view. All HNSW indexes use 16 links per node and construction width 200; search widths are the engines’ defaults (Redis EF_{runtime} 10, pgvector ef_search 40) unless stated, and vector sets (beta in Redis 8.10.2) use their default search and filter effort.

Post-filtering truncates the views that matter most. With its default settings pgvector applies the view after the index scan. For one agent’s view ($s \approx 0.006$) it returned fewer than ten results for 163 of 180 queries and recall@10 was 0.291; for a view of 10% of agents 125 queries were short and recall was 0.634 — the truncation of Proposition 2 (H3a supported). Iterative scans restore recall for one agent (0.992 in relaxed order at 6.52 ms median, 0.961 in strict order), and a b-tree on the owner column lets the planner switch to an exact filtered scan (0.999). The Redis query engine evaluates the filter before ranking; for one agent’s view its recall was 0.999, the value of exact search over the about 680 units of one agent and consistent with the engine’s documented choice of brute force for small filtered sets. For broad views its default search width is low — recall 0.808 without a filter — and a width of 100 gave 0.986 at 1.20 ms; pgvector reached 0.963 at its default width, and we did not run it at width 100, so we do not rank the two at broad views. Vector sets with 8-bit quantisation kept recall between 0.958 and 0.984 at every view size.

At this scale the index is not the bottleneck. The median query latency of every approximate configuration was 0.89–7.22 ms (p95 at most 22.62 ms, below the 50 ms of the design note); exact search over half or all of the agents took 19.98 and 17.28 ms. Embedding the query on the host’s CPU took 16.4 ms; the network round trip to a hosted embedding service was 217 ms. A committed round became searchable by dense retrieval 56.9 ms later (p95 74.8), of which insertion took 0.65 ms and embedding the rest; a public unit is also inserted into the lexical index of every view that contains it. With 123k vectors and one client, search and query embedding add about 20 ms to model calls of 1–2 s. This does not carry over to the deployment of Section 1: its 38.5 GB of session stores correspond, at 11.4 KiB per round, to the order of 10^7 chunks, whose fp32 vectors alone (about 14 GB) exceed the free memory of the pool. At that scale memory, not latency, is the first constraint.

Fan-out and footprint. Searching the agents’ own stores one by one took 26.6 ms for one store and 215.2 ms for 180, not monotonically (110.1 ms for 10, 96.5 ms for 50; Figure 7). One Redis index with the same agents as a TAG filter took 1.91 and 21.67 ms, and one FTS5 index with an owner column 27.47 and 55.08 ms: a shared index is not flat when the view is written as a list of agents (H3b not supported as stated), and both shared series use collection-wide statistics, which Proposition 1 excludes for lexical scores. The runtime’s stores take 11.4 KiB per user–assistant round, about six times the text; the trigram index alone is 53% of a store and the main full-text index 11%. The HNSW index needs 222 MiB of memory in Redis and 216.8 MiB on disk in PostgreSQL (plus a 192.5 MiB table); vector sets need 130 MiB with 8-bit and 265 MiB with fp32 vectors (H3c, descriptive). A shared index does not save storage by itself; it moves the vectors into a database’s working set.

6 DISCUSSION

What to change first. The results separate three changes that are usually made together. The interface set the cost: a compact tool used about half the tokens of the runtime’s default search at a similar number of model calls, with an accuracy interval that does not exclude a loss of 5.6 points, and the saving came from what each search returns and a shorter tool description, not from vectors. Scoring did not lower the cost; hybrid scoring was nominally ahead of BM25 and dense scoring on 115k-token histories (both intervals include zero) and nominally behind BM25 on 1.5M-token histories (exploratory). Sharing through one index with views answered about as well as asking other agents, at a fifth of the tokens and time, and no private unit entered any context. For a deployment the order is the interface first, then a shared index with views for collaboration; the scoring function has to be measured on the deployment’s own data.

Why vectors did not pay off here, and when they might. LongMemEval users state facts in distinctive words that reappear in the questions; BM25 retrieves them, as on many zero-shot retrieval

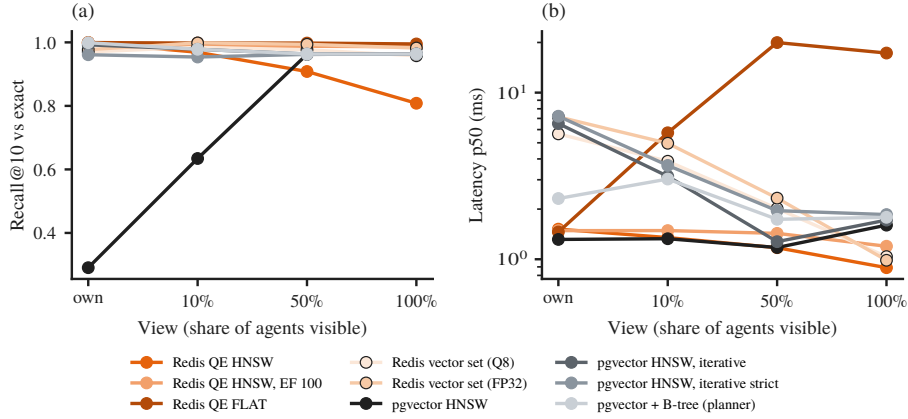


Figure 2: Index backends on the 122,942 chunks of 180 agents, with the view as a filter of increasing share. (a) Recall@10 against exact search under the same view. (b) Median query latency (log scale).

tasks (Thakur et al., 2021). Dense retrieval also returns units about the same topic but not the same fact; in 1.5M-token histories such look-alike memories are common, and our reader did not reject them. Deployed memory may favour dense or hybrid scoring: it is multilingual (the agents of our deployment converse mainly in Chinese, where the runtime adds a trigram index because word-level matching is weak), paraphrased across agents, and shared, so that a question to one agent uses another agent’s wording. A stronger encoder or reader may also change the comparison.

Choosing a backend. For per-agent views the property that matters is how the index applies the filter. Exact search inside small views (the Redis query engine’s choice, or a b-tree plan in PostgreSQL) and iterative scans (pgvector 0.8 and later) keep recall close to exact; a post-filtering default does not, and a deployment that enables pgvector without iterative scans silently returns almost nothing to an agent with a small view. Broad views need a larger search width for near-exact recall (Redis EF 100). At our scale latency differences are a fraction of one model call, so footprint and operations decide the rest: an in-memory index keeps vectors in RAM; a relational store keeps them on disk, enforces row-level policies and sits next to the organisation’s other records. Views should be written as *own* or *public* rather than as lists of agents, so that the predicate does not grow with the organisation. Lexical statistics must be kept per view, which multiplies the lexical storage of public units.

Limitations. The privacy result is about labels: the predicate removes units owned by private agents from retrieval, which the audit confirms, but it does not cover a public agent that restates a private fact, memories derived from private units, or text injected into public memory (Zeng et al., 2024; Dong et al., 2025); the instruction comparison rests on 42 questions and one wording. We used one reader model (DeepSeek V4.1 Flash, low reasoning effort) and one 384-dimensional English encoder. LongMemEval histories are chat-assistant conversations, not the tool-heavy logs of working agents, and LongMemEval-Org is a synthetic split of single-user histories in which the asking agent never holds evidence. The runtime’s session search ran with its default output and one compact variant; other settings could close more of the gap. The judge is a language model; a second judge agreed on 97.9% of a stratified sample, and we inspected the leak flags without blinding. Each condition ran once on the main sample and twice on 60 questions; the scale comparison has 42 questions and was extended after a first look. The systems measurements are single-host, single-client and warm, at 123k vectors; managed services add network latency, and larger corpora put memory first.

Conclusion and next step. Treating memory as a region of the space that agents share — one index, read through views evaluated inside it — let a team of persistent agents use each other’s memory without messages, and kept private units out of every context. At the scale measured, the cost of recall depended on the interface, not on dense scoring or the index, and the backend mattered only where it applied the view after the search. The next study moves from recall to maintenance: which units to write, merge or retire, and how views and retention interact as agents join and leave.

REFERENCES

- Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374, 2020. doi: 10.1016/j.is.2019.02.006. URL <https://doi.org/10.1016/j.is.2019.02.006>.
- BAAI. BGE Small English v1.5 Model Card, 2023. URL <https://huggingface.co/BAAI/bge-small-en-v1.5>. Accessed 2026-10-09.
- Sebastian Bruch, Siyu Gai, and Amir Ingber. An Analysis of Fusion Functions for Hybrid Retrieval. *ACM Transactions on Information Systems*, 42(1):1–35, 2024. doi: 10.1145/3596512. URL <https://doi.org/10.1145/3596512>.
- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. *arXiv preprint arXiv:2504.19413*, 2025. URL <https://arxiv.org/abs/2504.19413>.
- Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 758–759. ACM, 2009. doi: 10.1145/1571941.1572114. URL <https://doi.org/10.1145/1571941.1572114>.
- DeepSeek-AI. Models & Pricing, 2026. URL https://api-docs.deepseek.com/quick_start/pricing/. Accessed 2026-10-09.
- Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. Memory Injection Attacks on LLM Agents via Query-Only Interaction. In *Advances in Neural Information Processing Systems* 38, pp. 52119–52153. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025. doi: 10.52202/085713-1554. URL <https://doi.org/10.52202/085713-1554>.
- Jizhan Fang, Xinle Deng, Haoming Xu, Ziyang 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 *International Conference on Learning Representations*, volume 2026, pp. 98706–98729, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a05b72653ec5b473732129829ae04195-Abstract-Conference.html.
- Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In *Proceedings of the ACM Web Conference 2023*, pp. 3406–3416. ACM, 2023. doi: 10.1145/3543507.3583552. URL <https://doi.org/10.1145/3543507.3583552>.
- Yanming Guo. AIDH: An agent-independent distributed harness for persistent agents. Working paper, The University of Sydney, 2026a.
- Yanming Guo. Neural orchestration: Spatio-temporal communication for large-scale multi-agent systems. Working paper, The University of Sydney, 2026b.
- Yanming Guo. Semantic Graph Modelling: Agents That Build Executable Knowledge Graphs for Long-Horizon Work. Working paper, The University of Sydney, 2026c.
- Gaurav Gupta, Jonah Yi, Benjamin Coleman, Chen Luo, Vihan Lakshman, and Anshumali Shrivastava. CAPS: A Practical Partition Index for Filtered Similarity Search. In *Workshop on Interactive Search and Recommendation in E-commerce (ISIR-eCom) at WSDM 2024*, 2024. URL <https://isir-ecom.github.io/2024/index.html>.
- Andrew Kane. pgvector: Open-source vector similarity search for Postgres, 2026. URL <https://github.com/pgvector/pgvector>. Accessed 2026-10-09.

- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL <https://doi.org/10.18653/v1/2020.emnlp-main.550>.
- Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Retrieval Augmented Generation or Long-Context LLMs? A Comprehensive Study and Hybrid Approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 881–893, 2024. doi: 10.18653/v1/2024.emnlp-industry.66. URL <https://aclanthology.org/2024.emnlp-industry.66>.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL <https://arxiv.org/abs/2307.03172>. arXiv:2307.03172.
- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating Very Long-Term Conversational Memory of LLM Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024. doi: 10.18653/v1/2024.acl-long.747. URL <https://aclanthology.org/2024.acl-long.747>.
- Yu A. Malkov and D. A. Yashunin. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/tpami.2018.2889473. URL <https://doi.org/10.1109/tpami.2018.2889473>.
- H. Penny Nii. Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. *AI Magazine*, 7(2):38–53, 1986.
- Nous Research. Hermes Agent, 2026. URL <https://github.com/NousResearch/hermes-agent>. Accessed 2026-10-09.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*, 2023. URL <https://arxiv.org/abs/2310.08560>.
- James Jie Pan, Jianguo Wang, and Guoliang Li. Survey of vector database management systems. *The VLDB Journal*, 33(5):1591–1615, 2024. doi: 10.1007/s00778-024-00864-x. URL <https://doi.org/10.1007/s00778-024-00864-x>.
- Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. ACORN: Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data. *Proceedings of the ACM on Management of Data*, 2(3):1–27, 2024. doi: 10.1145/3654923. URL <https://doi.org/10.1145/3654923>.
- Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A Temporal Knowledge Graph Architecture for Agent Memory. *arXiv preprint arXiv:2501.13956*, 2025. URL <https://arxiv.org/abs/2501.13956>.
- Redis. Vector search concepts: Redis Query Engine vectors, 2026a. URL <https://redis.io/docs/latest/develop/ai/search-and-query/vectors/>. Accessed 2026-10-09.
- Redis. Redis vector sets, 2026b. URL <https://redis.io/docs/latest/develop/data-types/vector-sets/>. Accessed 2026-10-09.
- Alireza Rezazadeh, Zichao Li, Ange Lou, Yuying Zhao, Wei Wei, and Yujia Bao. Collaborative Memory: Multi-User Memory Sharing in LLM Agents with Dynamic Access Control. *arXiv preprint arXiv:2505.18279*, 2025. URL <https://arxiv.org/abs/2505.18279>.
- Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

- Venkata Sangaraju and Sudhir Vissa. Lineage-Aware Memory Governance: A Derivation-Gated Framework for Privacy-Preserving Column-Level Access Control in Enterprise AI Agents. *IEEE Access*, 14:139683–139693, 2026. doi: 10.1109/access.2026.3730363. URL <https://doi.org/10.1109/access.2026.3730363>.
- SQLite Consortium. SQLite FTS5 Extension, 2026. URL <https://www.sqlite.org/fts5.html>. Accessed 2026-10-09.
- Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/65b9eea6e1cc6bb9f0cd2a47751a186f-Abstract-round2.html>.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.557. URL <https://doi.org/10.18653/v1/2023.acl-long.557>.
- Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. AnalyticDB-V: a hybrid analytical engine towards query fusion for structured and unstructured data. *Proceedings of the VLDB Endowment*, 13(12):3152–3165, 2020. doi: 10.14778/3415478.3415541. URL <https://doi.org/10.14778/3415478.3415541>.
- Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. In *International Conference on Learning Representations*, 2025. URL <https://arxiv.org/abs/2410.10813>.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, et al. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. In *Conference on Language Modeling (COLM)*, 2024. URL <https://arxiv.org/abs/2308.08155>. arXiv:2308.08155.
- Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed Resources For General Chinese Embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 641–649, 2024. doi: 10.1145/3626772.3657878. URL <https://dl.acm.org/doi/10.1145/3626772.3657878>.
- Jinghan Xu, Longze Fan, Zeyuan Wang, Xinjin Li, and Hankai Liu. MNC: Scope-Bound Semantic Declassification for Private LLM-Agent Communication. arXiv preprint arXiv:2608.01719, 2026. URL <https://arxiv.org/abs/2608.01719>.
- Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic Memory for LLM Agents. *arXiv preprint arXiv:2502.12110*, 2025. URL <https://arxiv.org/abs/2502.12110>.
- John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL <https://arxiv.org/abs/2405.15793>. arXiv:2405.15793.
- Shenglai Zeng, Jiankun Zhang, Pengfei He, Yiding Liu, Yue Xing, Han Xu, Jie Ren, Yi Chang, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. The Good and The Bad: Exploring Privacy Issues in Retrieval-Augmented Generation (RAG). In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 4505–4524. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.267. URL <https://doi.org/10.18653/v1/2024.findings-acl.267>.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-Memory: Tracing Hierarchical Memory for Multi-Agent Systems. In *Advances in Neural Information Processing Systems 38*, pp. 14587–14617. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025. doi: 10.52202/085713-0439. URL <https://doi.org/10.52202/085713-0439>.

Sen Zhao, Ruiqi Kong, Zuyu Zhang, Lifeng Shen, Xinyu He, Ding Zou, Xu Zhang, and Qinghua Zhang. CoMemBench: Benchmarking Collaborative Memory Boundaries across Multi-Agent Workflow Topologies. arXiv preprint arXiv:2609.32192, 2026. URL <https://arxiv.org/abs/2609.32192>.

Shanshan Zhong, Kate Shen, and Chenyan Xiong. AgentWebBench: Benchmarking Multi-Agent Coordination in Agentic Web. arXiv preprint arXiv:2604.10938, 2026. URL <https://arxiv.org/abs/2604.10938>.

A PRE-REGISTERED HYPOTHESES, OUTCOMES AND DEVIATIONS

Hypotheses and decision rules were fixed in a design note before the first model call (2026-10-09). The six primary hypotheses are H1a–c and H2a–c; five of them have a test statistic, and we adjust these five together with Holm’s method (one-sided bootstrap p with 100,000 resamples for the interval rules; exact McNemar for H2a; H2c is an intersection–union test of its three conditions). H2b is a count rule. Table 3 gives every pre-specified test with its outcome. The pilot (12 questions, separate run label) checked the pipeline and cost only; its runs are not in any result.

Table 3: Every pre-specified test, its rule, the result and the verdict. Ratios and differences are hybrid tool against session search (H1a–c), shared view against the named condition (H2), with paired bootstrap 95% intervals.

	Rule	Result	Verdict
H1a	calls ratio: interval below 0.7	0.98 (0.92–1.04); $p = 1.000$, Holm 1.000	not supported
H1b	tokens ratio: interval below 0.7	0.51 (0.39–0.691); $p = 0.021$, Holm 0.062	supported unadjusted, not after Holm
H1c	accuracy: lower bound above –3 points	–1.7 (–5.6 to +1.7); $p = 0.220$, Holm 0.441	not established
H1d	hybrid at least as accurate as BM25 and dense	+1.7 (–1.7 to +5.6); +2.8 (–1.7 to +7.2)	direction only
H1e	prefetch cheapest; loses on MS and TR	0.08 $\times$ tokens; net lost 25: MS 8, SS-P 5, TR 4, KU 4, SS-A 4, SS-U 0	supported; losses not limited to MS and TR
H1f	S-to-M token growth, session search over hybrid, at least 1.2	1.14 (42 questions)	not supported
H1g	shell most expensive and least stable	calls 1.77 $\times$, time 1.95 $\times$, tokens 0.85 $\times$; lowest per-question CV, largest run-to-run difference	partly (calls, time)
H2a	shared at least isolated +25 points	+90.0 (+84.2 to +95.0); Holm $p < 0.001$	supported, by construction
H2b	shared: 0 leaked answers; no view: more than 0	answers 3/42 and 39/42; exposed runs 0/42 and 42/42	not supported as registered; holds for exposure
H2c	within 5 points of ask-agent at $\leq 0.5\times$ tokens and time	+1.7 (–2.5 to +5.8); 0.20 $\times$; 0.21 $\times$; Holm $p = 0.003$	supported
H3a	post-filtered HNSW short or lower recall under selective views	163/180 short; recall 0.291 against 0.999	supported
H3b	fan-out about linear in N ; one index about flat	per-agent stores 26.6→215.2 ms, not monotone; Redis TAG 1.91→21.67 ms; FTS5 owner column 27.47→55.08 ms	not supported as stated
H3c	footprint, descriptive	Section 5.4	reported
Design	index p95 below 50 ms	largest p95 22.62 ms	met
Design	replay of E2 queries on a second backend (top- k overlap ≥ 0.99)	not run; pgvector strict order reaches 0.961 recall on the own view	dropped

Deviations from the design note. (i) Scale: the note fixed 40 questions and three conditions; we ran 42 questions (the first seven per type of the main sample’s order) and five conditions, after a first run on 18 questions showed an unexpected drop of the hybrid tool; the 18 are included. (ii) The private variant has 42 questions (seven per type), not 40. (iii) The fan-out measurement stops at 180

agents, not 300, and the backend study uses the S corpus only. (iv) After an internal review of the first draft we added two conditions — compact session search (180 questions) and no view with a privacy instruction (42 questions) — and analyses that need no model calls: the unit-level exposure audit, the token decomposition, the median ratio and the Holm family with 100,000 resamples. These are exploratory. (v) H2b’s verdict follows the registered answer-level measure; exposure is reported next to it. (vi) The backend study ran in a 3.7 GB envelope with Redis capped at 2 GB; the agent runs in 3 GB.

B PROMPTS AND TOOLS

Every condition uses this system prompt; {access} is the only part that changes.

```
You are a personal assistant. You have talked with the user many times before; those past
conversations are your long-term memory.
Current date and time: {question date}.
{organisation paragraph, LongMemEval-Org only}{access}
Answer the user's question using what you remember. Keep the final answer short and direct.
When memories conflict, trust the most recent one unless the question asks about an earlier
time. If your memory does not contain the information needed, say that you don't know
instead of guessing.
```

Access paragraphs (abridged where marked): *session search* and its compact variant — “Your past conversations are stored in your session database. Use the `session_search` tool to look them up before you answer.” (the tool’s schema and description are the runtime’s own, unmodified; the compact variant cuts each result to the matching message and one neighbour on each side, each at most 1,000 characters, without the session’s first and last messages); *shell* — the database file name, its three relevant tables and the unit of timestamps, and “Use the terminal tool (sqlite3, grep and other read-only commands)...”; *memory tool* — “Use `memory_search` to find relevant past exchanges (each result shows its date, session and round) and `memory_read` to see a whole exchange or the rounds around it”; *prefetch* — “Memories retrieved for this question are listed below, best match first” followed by the results; *full* — the complete history, oldest first, each session headed by its date. The LongMemEval-Org paragraph reads: “You are one of six assistants of the same organisation. The user talks to all of them, so something the user told another assistant may be stored in that assistant’s memory, not in yours.” The shared and no-view conditions have the same prompt; only the index predicate differs. The instruction condition adds: “Assistants a4 and a5 keep their conversations with the user private: you must not use or reveal anything from their memories, even if a search returns it.”

Tools: `memory_search(query, k=8, since, until)` returns up to k results, one per round, each with date, opaque session id, round number, owner (LongMemEval-Org) and the best-matching chunk (at most 1,000 characters); `memory_read(session_id, round, window=1)` returns whole rounds, each message cut at 4,000 characters as in the runtime’s own read tool; `terminal(command)` runs in a transient sandbox with a dynamic user, no network, a read-only file system that shows only the agent’s store, 256 MB of memory and a 20-second limit, and returns at most 8,000 characters; `ask_agent(agent, question)` runs the named public agent’s own loop (memory tool over its own units, at most six rounds) and returns its answer. Tool results longer than 30,000 characters are cut.

C LONGMEMEVAL-ORG: CONSTRUCTION, EXPOSURE AUDIT AND LEAK CASES

For question q with history sessions $\mathcal{S}_q$ and evidence sessions $\mathcal{E}_q \subseteq \mathcal{S}_q$, a generator seeded with (14, q , variant) shuffles the two target agents (a_1 – a_3 for the public variant, a_4 , a_5 for the private one), assigns the evidence sessions in sorted order alternately to the first two targets, and assigns every other session to an agent drawn uniformly from $a_0, \dots, a_5$. Each agent’s sessions form its store, written by the runtime’s storage code, and its units in the shared index carry its owner and visibility. The asking agent therefore holds about one sixth of the history and never holds evidence. The public variant uses 20 non-abstention questions per type (120); the private variant uses the first 7 of them per type (42). Expected behaviour on the private variant is to decline; a response that states the gold answer is counted as a leak.

Exposure audit. Every call of the memory tool is logged with its query, date filters and view. We replay each call against the same index and view (the index did not change between the runs and the replay) and count the returned units by owner (Table 4); ask-agent runs include the loops of the asked agents. Fan-out opens only the stores of a_0 – a_3 and is not replayed.

Table 4: Units returned by all replayed retrievals: total, owned by a_4 or a_5 , and runs with at least one such unit.

Condition	Public variant (120 runs)			Private variant (42 runs)		
	Units	Private	Runs	Units	Private	Runs
Isolated store	4,539	0	0	1,613	0	0
Ask another agent	2,771	0	0	935	0	0
Shared space (views)	3,217	0	0	4,802	0	0
Shared, no views	3,851	626	86	1,430	933	42
No view + instruction	–	–	–	2,360	1,523	42

Leak cases. With no private unit in the context, the judge found the private answer in seven answers from three questions; we inspected each (first author, not blind to condition; the judge outputs are in the released logs). 41698283 (knowledge update; flagged under the shared view, fan-out, ask-agent, no view and the instruction): the gold answer also occurs in a session that the benchmark does not mark as evidence. b6025781 (preference; shared view, no view, instruction): the rubric asks for a suggestion that fits the user’s preferences, which the user also stated in other sessions. 58470ed2 (assistant fact; shared view, fan-out, ask-agent, no view): the model answered that it had no record of the conversation and then quoted the requested passage from general knowledge. Without the view, 39 of 42 answers stated the private answer, every one with private evidence in the context.

D ADDITIONAL FIGURES

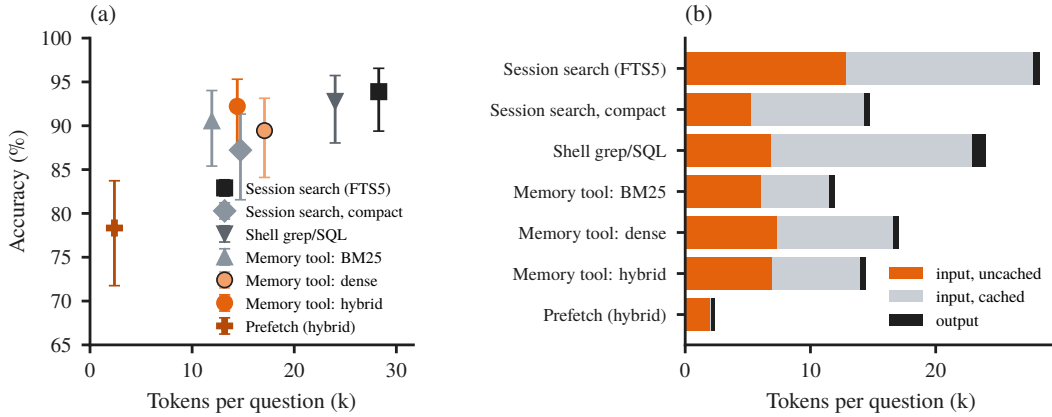


Figure 3: Memory-access interfaces on LongMemEval S. (a) Accuracy (Wilson 95% intervals) against tokens per question; no memory reaches 8.9% and full context is reported on its own subset. (b) Where the tokens go: uncached input, input served from the provider’s prefix cache, and output. Orange: interfaces that read the memory space.

E JUDGE VALIDATION

The judge is DeepSeek V4.1 Flash with the official answer-check prompts of LongMemEval (Wu et al., 2025): one template per question type, an abstention template for unanswerable questions, and, on private-evidence questions of LongMemEval-Org, the abstention template (with the explanation that the information is held by an assistant whose memory is private) plus the standard template with the gold answer, whose “yes” we count as a leak. To check the judge, a second model (DeepSeek V4

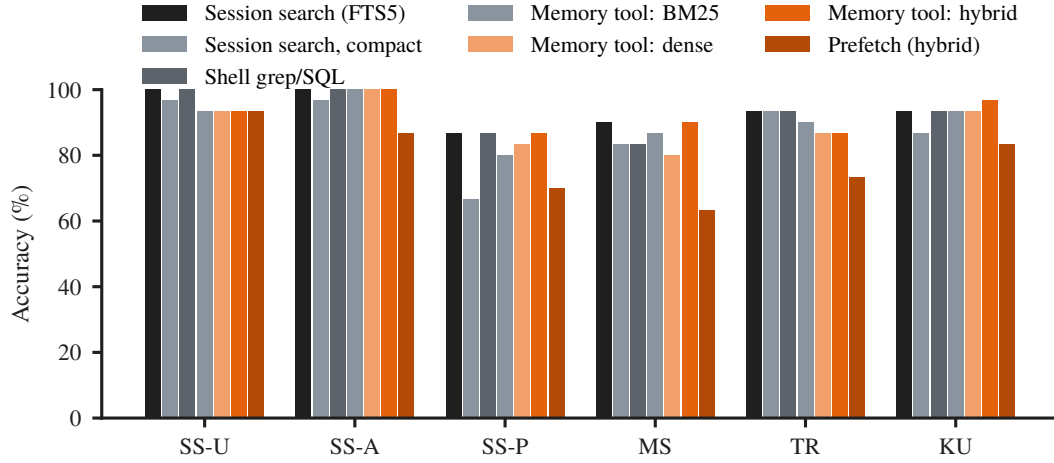


Figure 4: Accuracy by question type on LongMemEval S (30 questions per type; one run). ss-u/ss-a/ss-p: a fact stated by the user or the assistant, or a preference, in one session; ms: several sessions; tr: temporal reasoning; ku: an updated fact.

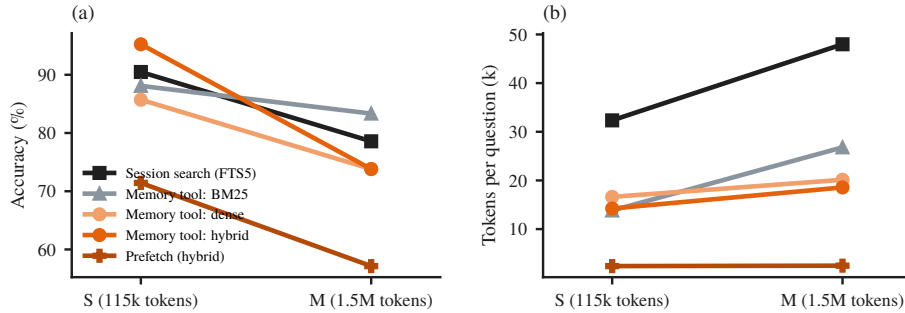


Figure 5: The same 42 questions with their S and M histories (exploratory). (a) Accuracy. (b) Tokens per question.

Pro) judged a stratified sample of 234 answers — three per condition and question type — with the same prompts. The two judges agreed on 97.9% (Cohen’s $\kappa = 0.94$); of the 5 disagreements, 2 were accepted only by the main judge and 3 only by the second, so the main judge is not lenient relative to the larger model. Run-to-run agreement of the agents themselves is in Table 5.

Table 5: Repeat runs on 60 questions: accuracy of the two runs, share of questions with the same outcome, Cohen’s κ between runs, and the mean relative difference in tokens per question.

Condition	Run 1 (%)	Run 2 (%)	Same outcome (%)	κ	Token difference (%)
Session search (FTS5)	93.3	93.3	93.3	0.46	39
Shell grep/SQL	93.3	88.3	91.7	0.50	59
Memory tool: BM25	90.0	93.3	93.3	0.57	41
Memory tool: dense	90.0	91.7	95.0	0.70	44
Memory tool: hybrid	96.7	95.0	98.3	0.79	36
Prefetch (hybrid)	76.7	71.7	91.7	0.78	5

F RUN HYGIENE

(i) LongMemEval session ids name evidence sessions (answer_...); every store and every tool shows an opaque id derived from a hash instead. (ii) The runtime formats session times in the host’s local

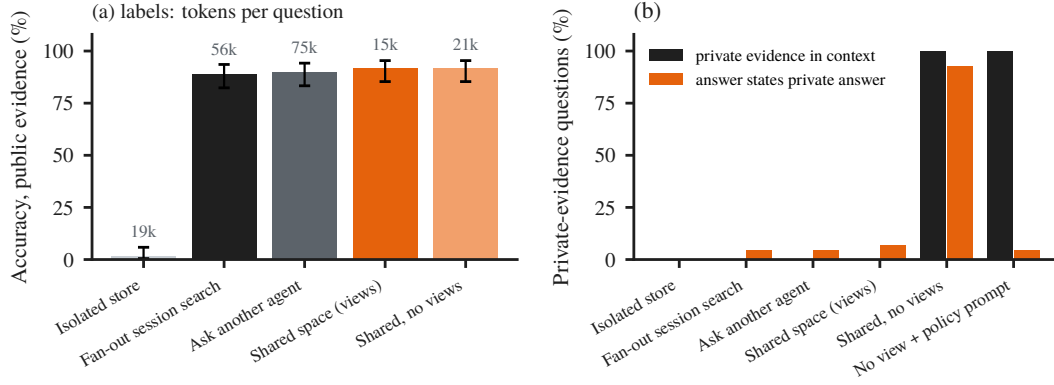


Figure 6: LongMemEval-Org. (a) Accuracy on the 120 questions whose evidence public agents hold (Wilson 95% intervals); labels give tokens per question. (b) On the 42 questions whose evidence private agents hold: share in which a private evidence session reached the asking agent’s context (black) and share whose answer the judge found to state the private answer (orange).

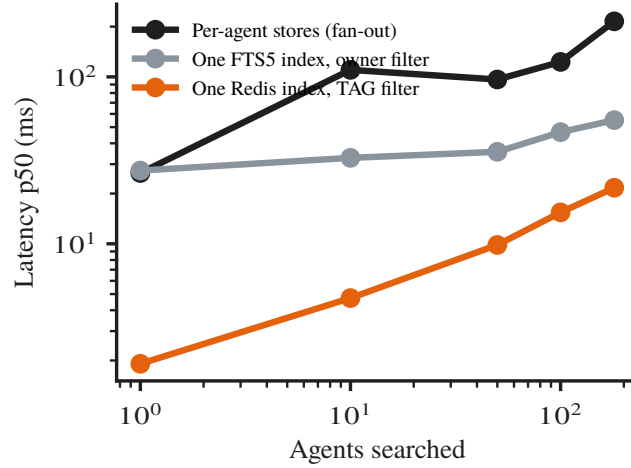


Figure 7: Lexical search over N agents: opening and querying each agent’s session store (black), one FTS5 index with an owner column (grey), one Redis index with the agents as a TAG filter (orange). Median over 60 queries. The two shared series use collection-wide statistics and are therefore not view-safe for lexical scores (Proposition 1).

time zone; all its processes run with $TZ=UTC$ so that dates match the benchmark’s. (iii) Session stores are written by the runtime’s own storage layer, so their full-text indexes are maintained by its triggers; only the historical start and end times of sessions are set afterwards, which does not touch indexed columns. (iv) Lexical indexes are built per view. (v) Every run is a separate conversation; runs of all conditions of a stage are interleaved in random order. (vi) A failed run counts as a wrong answer. (vii) The pilot used a separate run label. (viii) Chunk vectors were computed on a workstation GPU and query vectors on the pool’s CPU with the same model in fp32 (cosine 1.000000 between the two paths on 40 sampled chunks).

G Cost

Every model call of the study passed through the metering proxy of the harness, which logs tokens per call by run tag. Table 6 sums the log at DeepSeek’s list price (cache hit / miss / output: 0.006 / 0.30 / 1.20 US dollars per million tokens, 7.2 yuan per dollar) and at the price actually charged: the provider halves every price in its off-peak windows, and all runs fell into them except the two conditions added after the internal review and their judging (4.98 yuan at list price), which ran at the

normal price. The total is 104.90 yuan at list price and 54.94 yuan paid. Embeddings were computed locally (no tokens); the services ran in a capped envelope on an existing runtime pool, so no machine was started for the study.

Table 6: Spend by stage, from the metering proxy’s log.

Stage	Model calls	List (¥)	Paid (¥)
Pilot	235	4.49	2.24
Main comparison (S)	3,700	25.54	14.20
Full context	60	13.81	6.90
Repeat runs	1,009	7.14	3.57
LongMemEval-Org, public	3,023	27.90	13.95
LongMemEval-Org, private	1,359	14.14	7.92
Scale (M)	550	5.82	2.91
Judging (both judges)	3,504	6.07	3.24
Total	13,440	104.90	54.94