

Memory Under Noise: Why AI-Assistant Memory Engines Lose the Evidence as a User’s History Grows

Anonymous ACL submission

Abstract

We test five open-source memory engines for AI assistants (Hindsight, Mem0, Honcho, OpenViking, Graphiti) and three baselines on 1,000 curated questions about a person’s past conversations. As the person’s history fills with their own unrelated conversations (0, 50 and 90 % of what is ingested), we measure accuracy, cost, and where each answer’s evidence was lost. Model, embedder, prompt and judge are the same for all systems. A plain hybrid retrieval baseline (RAG) is the most accurate system at every level, 96.8 % at 90 % noise against 89.9 % for the best engine. Almost every error is a retrieval error. Reading the engines’ code and our traces, the main cause is how much each system returns. Systems that fill the context budget with relevance-ranked text stay robust. Systems that return a fixed number of items, or reserve slots for recent memories, lose the evidence as the store grows. Two findings correct earlier accounts of these runs. Honcho’s 90 % collapse is partly an unfinished build: every Honcho run at 90 % noise hit our 3-hour settle limit. And write cost grows with the number of sessions, not the number of tokens.

1 Introduction

Memory engines let an assistant remember its users. They use a language model to extract facts or observations at write time and return a compact context at question time (Packer et al., 2023; Chhikara et al., 2025; Rasmussen et al., 2025; Latimer et al., 2025). Their published evaluations mostly use small, clean histories, and each uses its own model, prompt and judge (Maharana et al., 2024; Wu et al., 2025). In real use, the part of a history that matters for any one question is a small fraction of everything the person has said.

We hold everything fixed except the memory system, and grow the irrelevant share of each person’s history from 0 to 50 to 90 %. We report three

findings, each traced to code and data:

- **RAG wins in our setting.** Chunked transcripts with hybrid BM25 + vector search beat every engine at every noise level, at a fraction of the cost (§3).
- **Errors are retrieval errors, and the retrieval policy decides them.** When the evidence reaches the answer model it is read correctly 94.5–98.8 % of the time. Whether it arrives depends on whether the system returns a fixed number of items or fills the budget with ranked text (§4).
- **Two corrections to our own earlier analysis.** Honcho was queried before its memory was built at 90 % noise, and write cost scales with sessions rather than tokens (§4, Appendix A).

2 Benchmark and setup

Questions. We took 200 conversations from LongMemEval (Wu et al., 2025), LoCoMo (Maharana et al., 2024), Multi-Session Chat (Xu et al., 2022) and Enron email (Klimt and Yang, 2004), 50 per corpus. An LLM agent read each one and wrote five questions at five difficulty levels, from recall to inference, with gold answers and evidence links. That gives 1,000 questions in 139 memory spaces, each one person’s history: a user, a pair of friends, or a mailbox. Details are in Appendix B.

Noise. Each space is ingested at 0, 50 and 90 % noise by tokens. Noise is the same person’s other history (mail, chats or sessions). When a LoCoMo or MSC pair runs out, it is new conversation generated in the pair’s voice, plus a few hard distractors. A model screens every noise session so that it cannot answer or contradict a question. The levels are nested prefixes of one noise list, so the signal is identical at every level and comparisons are paired by question. Ingested tokens grow from 280K to 2.80M.

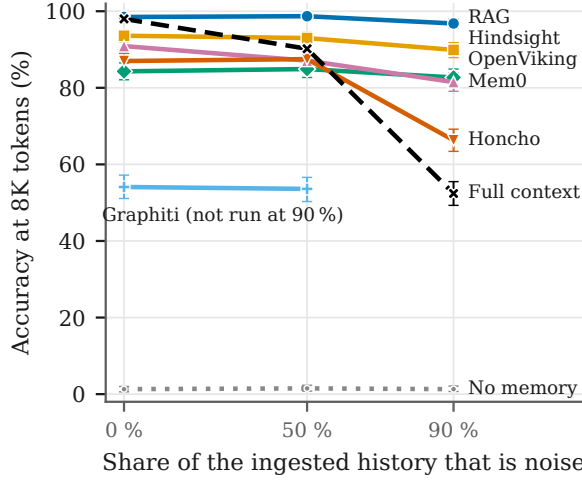


Figure 1: Accuracy against noise at an 8K budget, with 95 % intervals. Each line follows the same 1,000 questions as unrelated history is added.

Systems and controls. The engines run with their defaults (Vectorize, 2026; Mem0, 2026; Plastic Labs, 2026; Volcengine, 2026; Zep, 2026). The RAG baseline uses 350-token chunks, BM25 + cosine with reciprocal rank fusion (Cormack et al., 2009), and the top 40 chunks cut to the budget. Full context pastes the most recent sessions that fit; no memory is the floor. A tracing proxy forces one model on every call, including each engine’s internal calls: DeepSeek V4.1 Flash (DeepSeek, 2026) on one provider with minimal reasoning. Everything embeds with `text-embedding-3-small` (OpenAI, 2024). Each question is retrieved once at 8K tokens and answered with one prompt that says “don’t know” rather than guess.

Judging and statistics. One judge, Jev (Type-Safe, 2026), labels each answer. It also gives the probability that the retrieved context holds the gold answer’s evidence, which splits every error into a *retrieval miss* (evidence absent) or a *reading miss*. Against a second model judge it agrees on 98.3 % of 16,000 answers. Intervals are 95 % bootstrap intervals over questions (seed 7); differences are paired, and * marks an interval that excludes zero.

3 Results

Figure 1 and Table 1 give the main result. RAG is the most accurate system at every level. At 90 % noise it leads the best engine, Hindsight, by +6.9 [+5.0, +8.8] points. With no noise, every system except Graphiti scores 84.3–98.5 %; at 90 % the same systems span 52.4–96.8 %. Noise sorts them into three groups. RAG, Hindsight and

System	0 %	50 %	90 %	Δ 0→90	Cost
RAG	98.5	98.7	96.8	−1.7*	\$1.70
Hindsight	93.6	93.0	89.9	−3.7*	\$20.39
OpenViking	84.3	84.9	82.7	−1.6	\$20.73
Mem0	90.9	87.1	81.5	−9.4*	\$7.03
Honcho	87.0	87.5	66.3	−20.7*	\$18.12
Graphiti	54.1	53.6	—	−0.5 ^a	\$19.84 ^a
Full context	98.0	90.2	52.4	−45.6*	\$0.74
No memory	1.3	1.5	1.3	+0.0	\$0.16

Table 1: Accuracy (%) at 8K, paired change from 0 to 90 % noise (*: significant), and the total cost of the 90 % run. ^aGraphiti was run to 50 % only; its change and cost are at 50 %. Intervals and the 2K budget are in Appendix D.

OpenViking lose under four points; OpenViking’s change is not significant. Mem0 loses at each step. Honcho and full context hold until a threshold and then fall: Honcho by 21.2 points between 50 and 90 %, and full context as soon as the history no longer fits its window. Graphiti stays near 54 %. By dataset (Figure 2), LoCoMo is the hardest under noise, even for RAG. Honcho’s fall is broad. Mem0’s is concentrated on MSC persona chat (86.0 to 56.8 %), and full context survives only on MSC, whose histories stay short.

Errors are retrieval errors. Figure 3 splits the answers. Given the evidence, the shared answer model reads it correctly 94.5–98.8 % of the time for RAG and the four engines run at every level. Reading accuracy is lower only for Graphiti (90.1 %). What separates the systems is how often the evidence arrives at all. The commonest failure is a false “I don’t know”: at 90 % noise it is 28.1 % of Honcho’s answers, and stale or hallucinated answers are at most 0.3 % for any system.

Cost. Engines pay at write time (Figure 4). At 90 % noise, building memory cost the four extraction engines \$6.47–20.27, 86–98 % of their run cost. The whole RAG run cost \$1.70. From 0 to 90 % noise, ingested tokens grow 10× while build cost grows 11–22×.

4 Why systems lose the evidence

We read each engine’s retrieval code at the version we ran and matched it against what each system returned for every question. Table 2 summarises the result; Appendix A has the evidence.

Filling the budget with ranked text is robust. RAG returns its 40 best chunks and Hindsight runs

	Enron			LoCoMo			LongMemEval			MSC		
RAG	99	98	97	98	98	93	98	99	98	100	99	99
Hindsight	92	89	88	94	93	88	98	97	96	91	92	88
OpenViking	88	88	88	85	86	79	90	89	88	74	77	76
Mem0	95	96	93	88	90	81	94	93	96	86	69	57
Honcho	84	73	49	85	90	64	86	94	71	93	93	82
Graphiti	60	61	n/r	46	42	n/r	67	66	n/r	44	45	n/r
Full context	96	90	58	98	73	26	99	99	27	99	99	99
	0%	50%	90%	0%	50%	90%	0%	50%	90%	0%	50%	90%

Noise level within each dataset (accuracy at 8K, %; n/r: not run)

Figure 2: Accuracy (%) at 8K for every system, dataset and noise level (250 questions per cell). Darker is better. Intervals are in Table 8.

System	What the answer model receives (default)	Evidence present		Main cause of lost accuracy
		0 %	90 %	
RAG	top 40 raw chunks, cut to the budget	99.4	97.7	little to lose: ranked raw text fills the budget
Hindsight	four arms fused and reranked, cut to the budget	95.5	93.0	small losses; retrieval latency grows with the store
OpenViking	fixed quota: 3 events, 3 entities, 1 preference, ...	84.8	83.2	compact context misses evidence at every noise level
Mem0	top 20 memories by similarity, BM25 and entities	91.6	85.1	own conversation crowded out of the top 20; speaker names dropped at write time
Graphiti	top 10 facts	55.7	–	too few facts for multi-part answers
Honcho	≤100 observations per person: 33 similar, rest most recent	89.6	68.6	cap binds at 90 %; memory build unfinished at the 3-hour limit
Full context	most recent 8K tokens	98.6	55.7	evidence older than the window

Table 2: Retrieval policy, evidence presence (% , 8K) and root cause per system, from the engines’ code and our traces (Appendix A). Graphiti is at 55.3 % at 50 % noise.

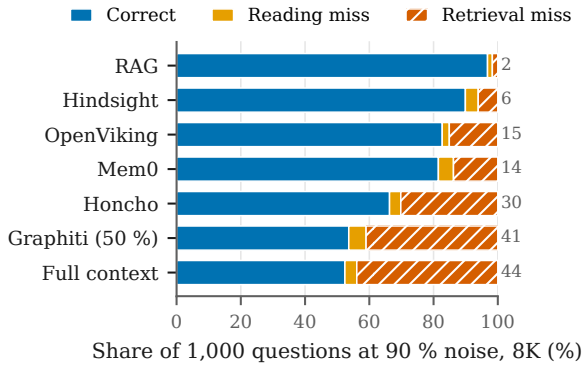


Figure 3: Every answer at 90 % noise (Graphiti at 50 %), 8K. Numbers on the right: retrieval misses, in percent. Reading misses are small for every system.

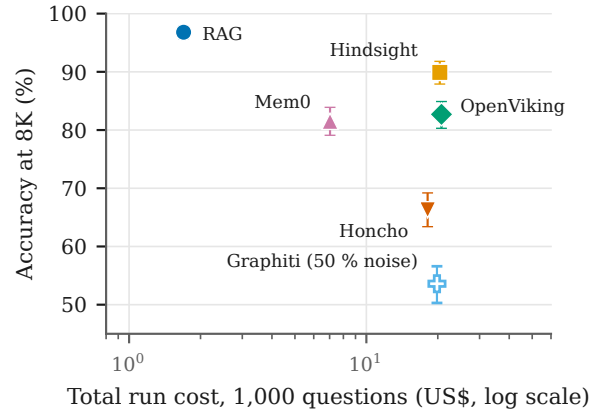


Figure 4: Accuracy against total run cost at 90 % noise (log scale), with 95 % intervals. Graphiti (hollow) is at 50 % noise, its highest level run.

four retrieval arms (semantic, BM25, graph, temporal) fused and reranked. Both fill the 8K budget, so a relevant item has room to stay in the context as the store grows (Figure 5; evidence present at 90 %: 97.7 and 93.0 %). Hindsight’s cost for this is latency: its median retrieval time grows from 2.8 s to 15.6 s.

A fixed number of items caps accuracy. Mem0 returns its top 20 memories and Graphiti its top 10 facts. OpenViking’s chat preset returns a fixed quota per category (3 events, 3 entities, 1 preference, ...), on average 6.9 entries and under 1,000 tokens, whatever the budget. OpenViking misses the evidence and answers wrongly on 12.8–15.1 %

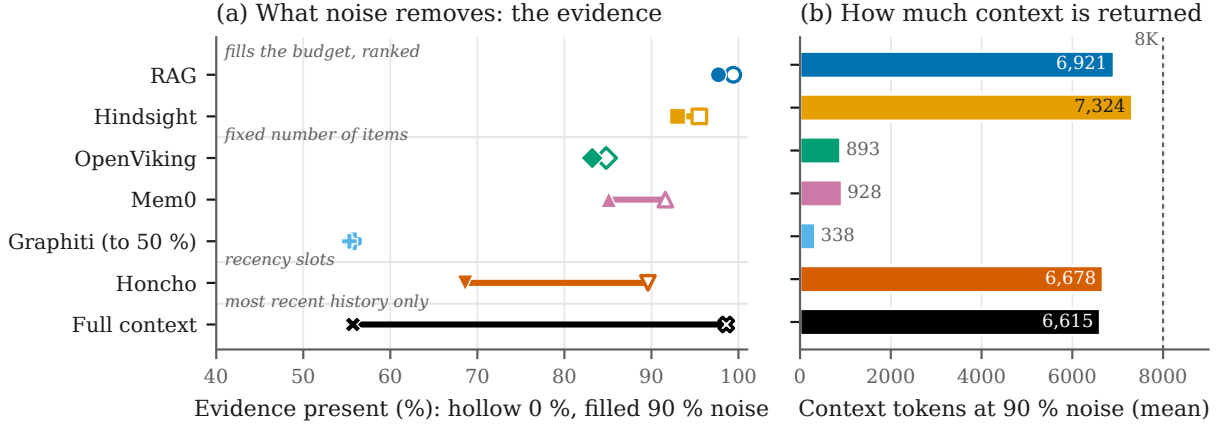


Figure 5: (a) Share of questions whose evidence reached the answer model, at 0 % (hollow) and 90 % (filled) noise, grouped by how each system decides what to return. (b) Mean context handed to the answer model at 90 % (Graphiti at 50 %). The two systems that fill the budget with relevance-ranked text keep their evidence. Fixed-size and recency-driven retrieval loses it as the store grows.

of questions at *every* noise level, so this is the cost of its compact context, not of noise. For Mem0 the cap starts to bind as noise grows. On MSC at 0 % fewer than 20 memories pass its similarity threshold (7.4 on average), so all of them come back. At 90 % only 19 % of the 20 come from the question’s own conversation, and 58 % come from noise about the same two people. Mem0 also drops speaker names: its extraction pipeline turns every message into `user: ...`, so 82 % of MSC memories read “User ...” and questions about who said what lose their anchor.

Honcho: recency slots, and an unfinished build.

Honcho’s context endpoint returns a “working representation” of at most 100 observations per person. A third are chosen by similarity to the question and the rest are the *most recent* observations. While a person’s store is small (LongMemEval and MSC at 0 and 50 %) the whole store fits, so noise costs nothing. At 90 % the cap binds for 95 % of LongMemEval questions, and two thirds of the slots go to the most recent observations, which at 90 % noise are mostly about noise. We also found that every Honcho run at 90 % noise reached our 3-hour settle limit with its deriver queue not drained (0 of 4 drained, against 4 of 4 at 50 %), so questions were answered from a partly built memory. At least 80 questions asked about conversations Honcho had not processed yet. They score 2.5 %, against 71.8 % for the rest (Figure 6). On LongMemEval the processed questions score 93.2 %, as high as at 50 % noise. Scoring the unprocessed questions at the processed rate would raise Honcho’s 90 % accuracy from 66.3 to about 73.1 %. The unfinished build

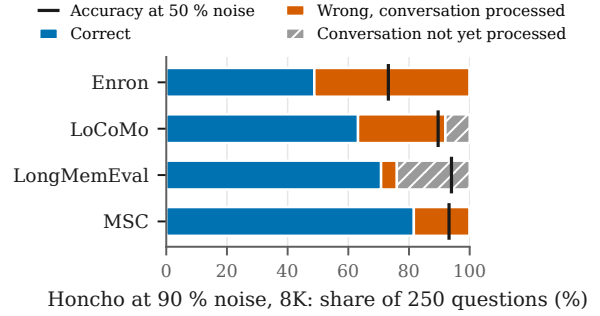


Figure 6: Honcho at 90 % noise. Grey: questions about conversations newer than anything in Honcho’s memory of that space when it was queried, a lower bound on unprocessed conversations. Black tick: accuracy at 50 % noise.

therefore explains part of the collapse, and the 100-observation cap is the likely cause of the rest. The 90 % number reflects Honcho under a fixed 3-hour build window, not Honcho’s capability.

Full context keeps only the most recent history.

It is the best system while a history fits in 8K tokens and fails as soon as the evidence is older than the last 8K. Only MSC, whose histories stay short, keeps it at 99.2 %.

Write cost follows sessions. The extraction engines make model calls per session with a roughly constant prompt (Appendix A.6). Noise sessions average 477 tokens against 1,399 for the curated conversations, so from 0 to 90 % the number of sessions grows 27× while tokens grow 10×. OpenViking and Mem0 calls grow 28× and 27×. Only Hindsight’s prompt per call also grows with its store (+41 %).

5 Discussion and limitations

What this does and does not show. In our setting, the deciding property of a memory system is whether its retrieval keeps the evidence as the store grows. In this setting, extraction and structure did not improve question answering over raw-text retrieval, and they cost write-time spend, detail and attribution. The setup favours RAG in several ways. A strong answer model reads raw text well. Most questions are anchored in one conversation, often with wording that overlaps the evidence. The noise is not adversarial. And several engines return far less context by default. The engines’ other benefits (compact, inspectable, editable memory) are not measured. The results do not rank engines in general. Changing only the provider routing and reasoning setting moved engines by -5.3 to $+12.5$ points (Appendix E).

Limitations. One seed per run, so intervals cover question sampling, not run-to-run variance. One model configuration and engine defaults, with no tuning; raising Mem0’s or Graphiti’s result counts was not tried. The questions were written by an LLM agent and checked by script, with no human audit, and one automatic judge was validated against another model but not against people. Some question types have 11–20 items. Most 90 % LoCoMo and MSC noise is generated (79 and 83 % of noise tokens), and none of it is adversarial. Honcho’s 90 % result is confounded by the settle limit, and Graphiti was not run at 90 %. The 2K budget is the first 2K tokens of each 8K result, which understates systems that return context in time order.

Ethics and reproducibility

All conversations come from public corpora (LongMemEval MIT; LoCoMo CC BY-NC 4.0; MSC via ParlAI (Miller et al., 2017); the Enron corpus is a public record). We add no personal data; generated distractors in Enron mailboxes are labelled as generated in the release. Models wrote the questions and the noise, and the questions are unaudited. The benchmark (content hash 4f18d65b4585b1742e12), every answer, verdict and model-call trace, and the scripts that generate every number, table and figure here (paper-v2/analysis/investigate.py, paper-v2/figures/make_figures.py) are released [TODO: repository link after review]. The engine source versions we read are listed in Ap-

pendix A.

References

- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. **Mem0: Building production-ready AI agents with scalable long-term memory**. *Preprint*, arXiv:2504.19413.
- Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. **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*, pages 758–759.
- DeepSeek. 2026. **DeepSeek V4.1 Flash**. <https://openrouter.ai/deepseek/deepseek-v4.1-flash>. Model deepseek/deepseek-v4.1-flash, released 10 September 2026; served through OpenRouter by DeepInfra (fp8).
- Bryan Klimt and Yiming Yang. 2004. The Enron corpus: A new dataset for email classification research. In *Machine Learning: ECML 2004*, volume 3201 of *Lecture Notes in Computer Science*, pages 217–226. Springer.
- Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. 2025. **Hindsight is 20/20: Building agent memory that retains, recalls, and reflects**. *Preprint*, arXiv:2512.12818.
- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. **Evaluating very long-term conversational memory of LLM agents**. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, Bangkok, Thailand.
- Mem0. 2026. **Mem0**. <https://github.com/mem0ai/mem0>. Python package mem0ai 2.2.1.
- Alexander H. Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. 2017. **ParlAI: A dialog research software platform**. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 79–84, Copenhagen, Denmark.
- OpenAI. 2024. **New embedding models and API updates**. <https://openai.com/index/new-embedding-models-and-api-updates/>. Introduces text-embedding-3-small.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. **MemGPT: Towards LLMs as operating systems**. *Preprint*, arXiv:2310.08560.

Plastic Labs. 2026. Honcho. <https://github.com/plastic-labs/honcho>. Server at commit 2eb27b6c; client honcho-ai 2.5.1.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. *Zep: A temporal knowledge graph architecture for agent memory*. Preprint, arXiv:2501.13956.

TypeSafe. 2026. Jev 1.13. <https://openrouter.ai/typesafe/jev-1.13>. Structured decision model typesafe/jev-1.13; see also <https://openrouter.ai/docs/guides/community/jev>.

Vectorize. 2026. Hindsight. <https://github.com/vectorize-io/hindsight>. Docker image ghcr.io/vectorize-io/hindsight:latest, pulled September 2026.

Volcengine. 2026. OpenViking: The context database for AI agents. <https://github.com/volcengine/OpenViking>. Docker image ghcr.io/volcengine/openviking:latest, pulled September 2026.

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

Xiaomi. 2026. MiMo-V2.6-Pro. <https://openrouter.ai/xiaomi/mimo-v2.6-pro>. Model xiaomi/mimo-v2.6-pro.

Jing Xu, Arthur Szlam, and Jason Weston. 2022. *Beyond goldfish memory: Long-term open-domain conversation*. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland.

Zep. 2026. Graphiti. <https://github.com/getzep/graphiti>. Python package graphiti-core 0.30.2.

A Root-cause investigations

For each result that needs explaining, we read the engine’s retrieval and write paths at the version we ran and checked the explanation against what the engine actually returned for every question. The returned items are stored per question in the published results (`retrieval.items_json`). All numbers here are generated by `paper-v2/analysis/investigate.py`.

Source versions read. Honcho at commit 2eb27b6 (the commit we ran); Mem0 at tag v2.2.1; Graphiti at tag v0.30.2. Hindsight and OpenViking ran as `:latest` Docker images pulled in September 2026, whose exact commits we did

Dataset	Settle 50 % (h)	Settle 90 % (h)	Drained	Empty	Unproc.	Acc. processed	Acc. 50 %	Acc. 90 %
Enron	0.67	3.00	no	0	0	48.8	73.2	48.8
LoCoMo	0.59	3.00	no	0	20	68.7	89.6	63.6
LongMemEval	0.87	3.00	no	30	30	93.2	94.0	71.2
MSC	0.43	3.00	no	0	0	81.6	93.2	81.6

Table 3: Honcho’s build (settle) time and outcome at 90 % noise. *Empty*: questions in spaces for which Honcho returned nothing at all. *Unproc.*: questions whose source conversation is newer than the newest observation Honcho returned for that space. *Acc. processed*: accuracy on the remaining questions.

not record. We read their source at the last commit before 25 September 2026 (Hindsight 32e66293b, OpenViking a86caca7), so line numbers for these two may differ slightly from what ran.

A.1 Honcho: recency slots and an unfinished build

What the context endpoint returns. Our adapter asks Honcho’s recommended context endpoint (`GET /sessions/{id}/context, src/routers/sessions.py:702`) for each person, with the question as `search_query`. The endpoint builds a *working representation* with `get_working_representation (src/crud/representation.py:317)`. That representation has at most `WORKING_REPRESENTATION_MAX_OBSERVATIONS = 100` observations (`src/config.py:945`). One third (`total // 3 = 33`) are chosen by semantic similarity to the question (line 340). The remainder are filled with the most recent observations (line 401). The merged list is then sorted by time (`src/utis/representation.py:398`). The token limit we pass does not shorten the representation, so what the answer model sees is at most 33 relevance-chosen observations per person plus the person’s latest ones.

Consequence: a threshold. While a person has fewer than 100 observations, the whole store is returned and noise costs nothing. Counting the timestamped observations Honcho returned per person, the cap was never reached on LongMemEval or MSC at 0 and 50 % (0 % of questions). It was reached for 95 % of LongMemEval and 66 % of MSC questions at 90 %. LoCoMo pairs, whose histories are long, are at the cap at every level (100 % already at 0 %). This matches Honcho being unaffected at 50 % and falling at 90 %.

The build did not finish. Honcho builds memory in a background derivier, and our runner waits for its queue to empty, for at most 3 hours. At

50 % noise the queue drained in at most 0.87 h. At 90 % all four runs stopped at the 3-hour limit and recorded "drained": false (Table 3). In the last 30 minutes before querying the deriver was still making 446–1,324 model calls per run, so the queue was not nearly empty. Questions were then answered from a partly built memory. Because the representation always includes a person’s most recent observations, the newest timestamp Honcho returns for a space shows how far its deriver had got. 6 LongMemEval spaces returned nothing at all (30 questions), and another 50 questions asked about conversations newer than anything Honcho returned for their space. These 80 questions score 2.5 % (Figure 6). This is a lower bound: the deriver need not process a space in time order, and on MSC the questions where Honcho returned fewer than 100 observations (likely incomplete stores) score lower than those at the cap.

How much each cause explains. On LongMemEval, the processed questions score 93.2 % at 90 % against 94.0 % at 50 %, so LongMemEval’s whole drop is the unfinished build. Across datasets, scoring the unprocessed questions at their dataset’s processed rate gives 73.1 % instead of 66.3 %. The rest of the fall, largest on Enron mailboxes (48.8 % with no detected unprocessed questions), coincides with the 100-observation cap. In mailboxes our adapter reads the owner’s representation plus the owner’s view of every correspondent, each capped separately. We did not rerun Honcho with a longer settle (no new runs were allowed), so the size of the remaining cap effect is an inference.

A.2 Mem0: a fixed top-20 and lost speaker names

Retrieval. Our adapter calls `Memory.search(query, top_k=20)` with the default threshold 0.1. Mem0 combines vector similarity, BM25 and an entity boost (`ENTITY_BOOST_WEIGHT = 0.5`, `mem0/utils/scoring.py:57`) and keeps the top 20. Returned memories per question grow from 13.2 at 0 % to the cap, 20.0, at 90 %. Each returned memory records the session it was extracted from, so we can tell where the context came from (Figure 7). On MSC, 100 % of returned memories come from the question’s own conversation at 0 %, 49 % at 50 % and 19 % at 90 %. At 90 %, 58 % come from generated conversation between the same two people, which

shares their topics and personas. Over all datasets the own-conversation share falls from 60 to 27 %. The evidence is crowded out of a fixed-size list.

Speaker names. Our adapter passes each message’s speaker as `name`. Mem0’s extraction path formats messages with `parse_messages` (`mem0/memory/utils.py:61`), which writes `user: <content>` and drops the name. In a two-person chat both people become “user”, and names reach a memory only if they appear in the text. They appear in 0.0 % of real MSC messages. As a result 82 % of Mem0’s MSC memories at 0 % noise read “User ...”, which loses who said what for questions such as “Which of the two is vegan?”. Each `add` also makes one extraction call that includes the 10 most similar existing memories (`mem0/memory/main.py:931`), which is why its calls grow with sessions (Appendix A.6).

A.3 OpenViking: a fixed quota per category

OpenViking’s `search` in `context` mode assembles a context from category buckets. With `purpose="chat"` (our setting) the bucket ceilings are 3 events, 3 entities, 1 preference, 1 experience, 1 resource and 1 skill (`PURPOSE_PRESETS`, `openviking/retrieve/context_assembler/params.py:81–97`). Each entry is served at a summary tier, and only events may deepen when budget is left over. The `max_tokens` we pass therefore bounds the context but does not fill it. OpenViking returned 6.2–6.9 entries per question at every noise level, one of them a fixed resource-directory description, for 627–893 tokens. Since the quota is the same with and without noise, so is the miss rate. That is why OpenViking is flat under noise but stays +14.1 [+11.6, +16.6] points below RAG at 90 %.

A.4 Graphiti: ten facts

Graphiti’s default `search` returns `DEFAULT_SEARCH_LIMIT = 10` edges (`graphiti_core/search/search_config.py:29`), each a one-sentence fact; we returned them with their validity intervals, about 323 tokens. Ten facts cannot cover multi-part answers: Graphiti scores 30.0 % on L3 synthesis questions against 80.5 % on L1 recall, and its evidence is present for 55.7 % of questions. Its store is not the limit: raising the result count was considered but not run.

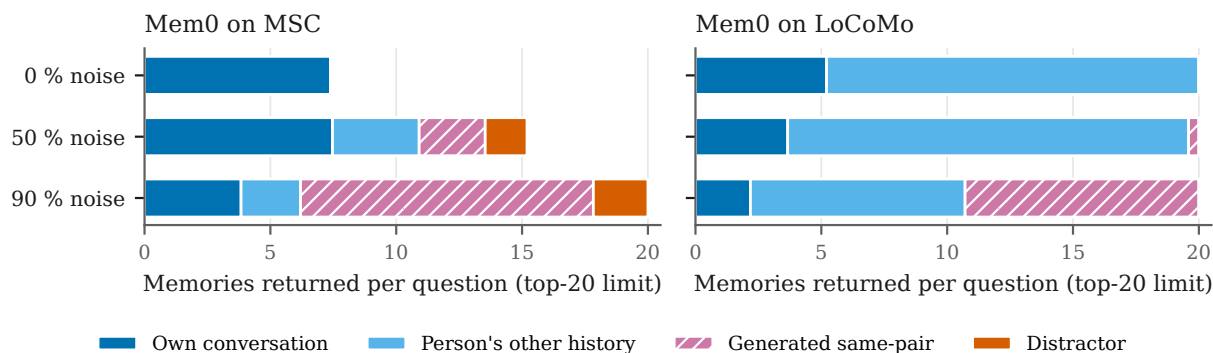


Figure 7: Where Mem0’s returned memories come from, per question at 8K (top-20 limit). As noise grows, memories from the question’s own conversation are pushed out by the same people’s other and generated conversations.

Engine	Calls 0 %	Calls 90 %	Growth	Prompt/call 0 %	Prompt/call 90 %	Cost growth
Hindsight	914	13,882	15.2×	5,063	7,151	18.7×
OpenViking	405	11,477	28.3×	5,593	5,318	19.9×
Mem0	205	5,477	26.7×	9,557	9,429	21.9×
Honcho	1,092	15,113	13.8×	3,895	4,493	11.4×
Graphiti (to 50 %)	15,293	32,869	2.1×	1,608	1,756	2.2×

Table 4: Main-model calls in the build phases (ingest + settle) from the traces, mean prompt tokens per call, and growth of build cost. Graphiti is shown to 50 %.

of tokens. Prompt tokens per call change by -1% for Mem0, -5% for OpenViking and $+15\%$ for Honcho. Only Hindsight’s prompt grows with its store (5,063 to 7,151 tokens, $+41\%$), which adds to its call growth. For a deployed assistant, write cost therefore depends on how many conversations a user has, not only on their length.

A.5 Hindsight, RAG and full context

Hindsight’s `recall` retrieves along four arms (semantic, BM25, graph and temporal), fuses them with reciprocal rank fusion and reranks with a local cross-encoder (`hindsight_api/engine/memory_engine.py`, recall ranking strategies at lines 670–677). It then fills `max_tokens`: 7,324 tokens at 90 %. Like RAG, it hands the answer model a long, relevance-ranked context, and both keep their evidence (93.0 and 97.7 % present at 90 %). We did not profile its retrieval, whose median latency rises from 2.8 s to 15.6 s as the store grows. Full context returns the most recent sessions that fit 8K tokens by construction, so it fails as soon as the evidence is older than that window.

A.6 Why build cost grows faster than the input

An earlier reading of these runs attributed the faster-than-input growth of build cost to consolidation work that grows with the store. The traces show a simpler main cause (Figure 8, Table 4). Engines extract per session (Mem0 and OpenViking) or per batch of messages (Honcho), with a prompt of roughly constant size. Noise sessions are short (477 tokens on average, against 1,399 for the curated conversations), so the number of sessions, and of extraction calls, grows much faster than the number

B Benchmark details

Curation. The main model first scored sampled conversations for density of specific content, which only set the reading order. An LLM agent then read each shortlisted conversation (the curation record lists the reviewer as “claude”). It accepted those with enough independent, self-consistent detail for five questions (11 of 211 rejected, with reasons) and wrote one question per level: L1 recall, L2 paraphrase or disambiguation, L3 synthesis, L4 time or attribution, and L5 deep (inference, preference, change over time, or an unanswerable question). A script checks that each question’s evidence lies in its space and shares words with the gold answer. Questions are asked after their whole space is ingested, on average days to months after the source conversation (Table 5).

Noise. Candidate noise is screened by the main model against all of the space’s questions, and dropped if it states, implies, contradicts or updates an answer, discusses the same specific event, person or number, or makes an unanswerable question answerable (876 of 6,155 dropped). Generated same-pair conversation is used only when a pair runs out of its own history. At 90 % that is most LoCoMo and MSC noise. 5 spaces reach 88.0–89.7 % instead of 90 %.

Example (Enron, one mailbox, thread about an ISDA agreement with the City of Glendale). L1:

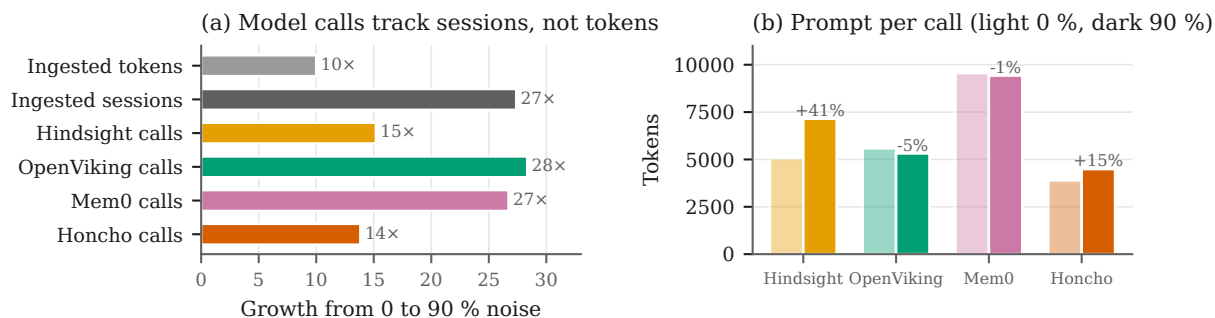


Figure 8: (a) From 0 to 90 % noise, ingested tokens grow 10× but sessions grow 27×, and the engines’ build calls track sessions. (b) Prompt tokens per build call are nearly constant, except for Hindsight.

Dataset	Spaces	Conv. tok.	Gap (d)	Signal	Noise 50/90 %	Sessions 0/50/90 %	90 % T2/T2G/T3 (%)	Screened (dropped)
Enron	29	1,353	75	68K	71K / 616K	50 / 345 / 2,403	98 / 0 / 2	2,459 (4 %)
LoCoMo	10	836	106	42K	42K / 372K	50 / 122 / 1,464	21 / 79 / 0	1,677 (16 %)
LongMemEval	50	3,018	19	151K	152K / 1.36M	50 / 121 / 506	100 / 0 / 0	491 (7 %)
MSC	50	388	10	19K	20K / 173K	50 / 175 / 1,104	11 / 83 / 5	1,528 (31 %)
Total	139			280K	284K / 2.52M	200 / 763 / 5,477		6,155 (14 %)

Table 5: Benchmark and suite statistics. *Conv. tok.*: mean tokens of a curated conversation. *Gap*: mean days between a question’s source conversation and the time it is asked. *Sessions*: events ingested per suite. *T2/T2G/T3*: share of 90 %-noise tokens that is the person’s own other history, generated same-pair conversation, or a hard distractor. *Screened*: candidate noise sessions checked against the space’s questions, and the share dropped.

Who was the City of Glendale’s attorney I dealt with on the ISDA? (Steve Lins, Assistant City Attorney). L3: How many legal opinions was ENA requiring, and who was paying for which? (two: one in-house, one from outside counsel that ENA paid for). L5: Could we document a Glendale foreign-exchange deal, or a power deal expiring in December 2002, under the executed ISDA? (no to both).

C Setup details

Systems. Hindsight: each session retained as one document, one bank per space; recall with `max_tokens = budget` and budget level “mid”. Mem0 (OSS, pgvector, 2.2.1): add per session with the date in-band; `search(top_k=20)`. Honcho (commit 2eb27b6): messages with timestamps, one peer per speaker, every mailbox participant observing the others; settle until the deriver queue is empty (at most 3 h); the context endpoint per person. OpenViking: per-session commit and wait; search in context mode, purpose “chat”. Graphiti (0.30.2, Neo4j): one episode per message; default search (10 facts). Three throughput changes from defaults: Honcho’s deriver uses 8 workers and a 60 s batch flush, and OpenViking’s model-client timeout is 1,800 s at 90 % noise.

Models and judging. DeepSeek V4.1 Flash for every role except embedding and judging, pinned to one provider (DeepInfra, fp8), minimal reasoning, 8,000 tokens of reasoning headroom and a 16,384-token output cap. The judge, Jev, returns one of six verdicts (correct, abstain-correct, wrong, abstain-wrong, stale, hallucinated) and the probability that the context holds the evidence (present if ≥ 0.5). On 16,000 answers it agreed with MiMo-V2.6-Pro (Xiaomi, 2026) on correct versus not for 98.3 %, and on evidence presence for 96.4 %.

Answer prompt (identical for every system; {who} names the pair or the mailbox owner):

```
You are an assistant with long-term memory of past
conversations{who}.
Answer the question using the memories provided.
Rules:
- The current date is {now}. Resolve relative dates
("last week", "two months ago") against the memory
dates.
- If memories conflict, prefer the most recent one
unless the question asks about an earlier time.
- If the memories do not contain the information,
say that you don't know. Do not guess.
- Answer concisely: one or two sentences, stating
the specific fact asked for.
<memories> - [date] text ... </memories>
Question: {question}
```

The full judge instructions are in the repository (bench/judge_alt.py).

System	0 % noise	50 % noise	90 % noise
<i>8K-token budget</i>			
RAG	98.5 [97.7, 99.2]	98.7 [97.9, 99.3]	96.8 [95.7, 97.9]
Hindsight	93.6 [92.0, 95.0]	93.0 [91.4, 94.5]	89.9 [87.9, 91.8]
OpenViking	84.3 [82.1, 86.5]	84.9 [82.7, 87.1]	82.7 [80.3, 84.9]
Mem0	90.9 [89.1, 92.7]	87.1 [85.1, 89.1]	81.5 [79.1, 83.9]
Honcho	87.0 [84.8, 89.0]	87.5 [85.5, 89.5]	66.3 [63.4, 69.2]
Graphiti	54.1 [51.1, 57.2]	53.6 [50.3, 56.6]	<i>not run</i>
Full context	98.0 [97.0, 98.8]	90.2 [88.3, 92.0]	52.4 [49.3, 55.5]
No memory	1.3 [0.6, 2.0]	1.5 [0.8, 2.3]	1.3 [0.7, 2.1]
<i>2K-token budget</i>			
RAG	97.1 [96.0, 98.1]	95.1 [93.8, 96.4]	91.7 [90.0, 93.3]
Hindsight	93.5 [92.0, 95.0]	92.8 [91.2, 94.3]	89.5 [87.5, 91.3]
OpenViking	84.3 [81.9, 86.5]	84.8 [82.5, 86.9]	81.2 [78.9, 83.6]
Mem0	91.0 [89.1, 92.7]	87.5 [85.5, 89.5]	82.1 [79.8, 84.5]
Honcho	67.2 [64.2, 70.2]	58.1 [55.3, 61.2]	35.9 [33.0, 38.9]
Graphiti	54.6 [51.5, 57.6]	53.7 [50.7, 56.8]	<i>not run</i>
Full context	72.3 [69.4, 75.2]	47.3 [44.1, 50.3]	11.3 [9.4, 13.2]
No memory	1.2 [0.6, 1.9]	1.3 [0.6, 2.0]	1.5 [0.8, 2.2]

Table 6: Accuracy (%) with 95 % bootstrap intervals. The 2K results for Honcho and full context understate them: both return context in time order, and the 2K context is the first 2,000 tokens of the 8K result.

System	0 → 50 %	50 → 90 %	0 → 90 %
RAG	+0.2 [-0.5, +0.9]	-1.9* [-2.8, -1.0]	-1.7* [-2.9, -0.7]
Hindsight	-0.6 [-2.2, +0.9]	-3.1* [-4.8, -1.3]	-3.7* [-5.6, -2.0]
OpenViking	+0.6 [-1.7, +2.8]	-2.2 [-4.5, +0.1]	-1.6 [-3.9, +0.7]
Mem0	-3.8* [-6.3, -1.3]	-5.6* [-8.2, -3.0]	-9.4* [-12.0, -6.7]
Honcho	+0.5 [-2.3, +3.4]	-21.2* [-24.5, -18.1]	-20.7* [-24.0, -17.4]
Graphiti	-0.5 [-3.0, +2.1]	-	-
Full context	-7.8* [-9.6, -6.0]	-37.8* [-40.8, -34.6]	-45.6* [-48.9, -42.4]
No memory	+0.2 [+0.0, +0.5]	-0.2 [-0.5, +0.0]	+0.0 [+0.0, +0.0]

Table 7: Paired change in accuracy (points) between noise levels at 8K; *: significant.

D Full results

All tables come from scripts/report_analysis.py run on the published results (2,000 bootstrap resamples, seed 7); they are shared with the long version of this paper.

E Sensitivity to the model setting

Before the reported runs, the no-noise stage ran with the main model spread by the router over about 25 providers and with reasoning off (Table 15). The baselines gained about 2 points from minimal reasoning alone, while the engines moved by -5.3 to +12.5 points. OpenViking’s extraction produced far more memories about other people (Enron +40.4), and Graphiti’s facts became more literal. Some providers returned corrupted text: one engine’s MSC store held observations such as “The

System	Enron	LoCoMo	LongMemEval	MSC
RAG	98.8 / 98.4 / 97.2	98.0 / 98.0 / 92.8	97.6 / 99.2 / 98.0	99.6 / 99.2 / 99.2
Hindsight	91.6 / 89.2 / 88.4	94.4 / 93.2 / 88.0	97.6 / 97.2 / 95.6	90.8 / 92.4 / 87.6
OpenViking	88.4 / 88.0 / 88.0	85.2 / 85.6 / 78.8	90.0 / 89.2 / 88.4	73.6 / 76.8 / 75.6
Mem0	95.2 / 96.0 / 92.8	88.4 / 90.4 / 80.8	94.0 / 93.2 / 95.6	86.0 / 68.8 / 56.8
Honcho	83.6 / 73.2 / 48.8	84.8 / 89.6 / 63.6	86.4 / 94.0 / 71.2	93.2 / 93.2 / 81.6
Graphiti	59.6 / 61.2 / -	45.6 / 42.4 / -	67.2 / 65.6 / -	44.0 / 45.2 / -
Full context	95.6 / 90.0 / 58.0	98.4 / 72.8 / 25.6	99.2 / 98.8 / 26.8	98.8 / 99.2 / 99.2

Table 8: Accuracy (%) at 8K by dataset, at 0 / 50 / 90 % noise (250 questions per cell).

Ivan stated that they accepted their last English test”. Engine results are therefore only comparable under one fixed model configuration.

F Corrections to the earlier report

Writing this paper from the repository’s full report (docs/REPORT.md) and the data, we found these differences; the paper follows the data.

1. Every Honcho run at 90 % noise ended its settle phase at the 3-hour limit without draining (Appendix A.1). The report presents the 90 % result as a retrieval failure.
2. Build cost grows faster than the input mainly because sessions grow faster than tokens, not because of consolidation (Appendix A.6).
3. At 90 % noise, 79 % of LoCoMo and 83 % of MSC noise tokens are generated conversation, not the pair’s own history.
4. At a 2K budget and 90 % noise, RAG and Hindsight tie (+2.2 [-0.1, +4.7]).
5. Reading accuracy of 94–99 % holds for RAG and four engines, not for Graphiti (90.1 %) or full context at 90 % (93.4 %).
6. The questions were written by an LLM agent; no human review of them is recorded.
7. LoCoMo is licensed CC BY-NC 4.0.

Row — column	Hindsight	OpenViking	Mem0	Honcho	Full context
RAG	+6.9* [+5.0, +8.8]	+14.1* [+11.6, +16.6]	+15.3* [+12.7, +17.8]	+30.5* [+27.5, +33.6]	+44.4* [+41.2, +47.7]
Hindsight		+7.2* [+4.4, +9.7]	+8.4* [+5.6, +11.1]	+23.6* [+20.3, +27.0]	+37.5* [+33.9, +41.1]
OpenViking			+1.2 [-1.8, +4.1]	+16.4* [+12.8, +20.0]	+30.3* [+26.3, +34.2]
Mem0				+15.2* [+11.6, +18.6]	+29.1* [+24.9, +33.4]
Honcho					+13.9* [+9.8, +18.0]

Table 9: System versus system at 90 % noise, 8K: row minus column (points), paired 95 % intervals.

System	Noise	Evid. present	Reading acc.	Retr. miss	Read. miss
RAG	0 %	99.4	98.8	0.3	1.2
	50 %	99.9	98.7	0.0	1.3
	90 %	97.7	98.5	1.7	1.5
Hindsight	0 %	95.5	97.4	3.9	2.5
	50 %	94.8	97.0	4.2	2.8
	90 %	93.0	95.7	6.1	4.0
OpenViking	0 %	84.8	97.9	13.9	1.8
	50 %	85.2	97.3	12.8	2.3
	90 %	83.2	97.4	15.1	2.2
Mem0	0 %	91.6	97.2	6.5	2.6
	50 %	89.8	95.4	8.8	4.1
	90 %	85.1	94.5	13.8	4.7
Honcho	0 %	89.6	96.7	10.0	3.0
	50 %	89.3	97.1	9.9	2.6
	90 %	68.6	94.9	30.2	3.5
Graphiti	0 %	55.7	90.1	40.4	5.5
	50 %	55.3	90.2	41.0	5.4
Full context	0 %	98.6	99.2	1.2	0.8
	50 %	90.6	99.1	9.0	0.8
	90 %	55.7	93.4	43.9	3.7
No memory	0 %	3.5	34.3	96.4	2.3
	50 %	3.8	31.6	95.9	2.6
	90 %	3.9	30.8	96.0	2.7

Table 10: Retrieval versus reading at 8K (%). Reading accuracy: accuracy where the evidence was present.

System	Noise	L1 recall	L2 paraphr.	L3 synth.	L4 time/who	L5 deep
RAG	0 %	99.5	99.5	97.5	98.5	97.5
	50 %	100.0	99.5	97.0	99.5	97.5
	90 %	99.0	98.0	94.0	98.0	95.0
Hindsight	0 %	98.0	94.5	90.0	96.5	89.0
	50 %	98.0	93.5	88.0	96.5	89.0
	90 %	96.0	91.5	83.5	93.0	85.5
OpenViking	0 %	95.0	88.5	78.0	83.0	77.0
	50 %	94.5	90.0	75.5	83.0	81.5
	90 %	92.0	85.5	74.5	84.5	77.0
Mem0	0 %	92.5	92.5	90.0	87.5	92.0
	50 %	91.0	87.5	84.5	86.0	86.5
	90 %	84.0	83.0	78.5	80.0	82.0
Honcho	0 %	91.0	86.5	81.5	88.0	88.0
	50 %	95.0	88.0	81.0	89.0	84.5
	90 %	74.5	64.0	58.5	69.0	65.5
Graphiti	0 %	80.5	61.0	30.0	51.5	47.5
	50 %	79.0	61.5	26.5	51.5	49.5
Full context	0 %	99.0	98.5	97.0	97.5	98.0
	50 %	91.5	91.5	89.5	89.5	89.0
	90 %	52.5	51.5	51.0	53.0	54.0
No memory	0 %	0.0	0.0	0.0	0.0	6.5
	50 %	0.0	0.0	0.0	0.0	7.5
	90 %	0.0	0.0	0.0	0.0	6.5

Table 11: Accuracy (%) by difficulty level at 8K (200 questions per level).

System	Noise	Build (h)	Build calls	Build \$	\$/1K tok.	Answer \$	Total \$	¢/correct
RAG	0 %	0.0	200 ^e	\$0.01	\$0.0000	\$0.47	\$0.73	—
	50 %	0.1	763 ^e	\$0.01	\$0.0000	\$0.85	\$1.21	—
	90 %	0.4	5,477 ^e	\$0.06	\$0.0000	\$1.18	\$1.70	0.18
Hindsight	0 %	0.5	6,016	\$1.00	\$0.0036	\$0.43	\$1.67	—
	50 %	1.5	13,606	\$2.56	\$0.0045	\$0.71	\$3.60	—
	90 %	11.1	76,272	\$18.70	\$0.0067	\$1.18	\$20.39	2.27
OpenViking	0 %	0.9	5,424	\$1.02	\$0.0036	\$0.25	\$1.39	—
	50 %	3.2	11,337	\$2.87	\$0.0051	\$0.26	\$3.26	—
	90 %	21.2	63,817	\$20.27	\$0.0072	\$0.31	\$20.73	2.51
Mem0	0 %	0.9	808	\$0.30	\$0.0011	\$0.29	\$0.72	—
	50 %	1.6	3,224	\$0.93	\$0.0016	\$0.37	\$1.44	—
	90 %	9.6	21,665	\$6.47	\$0.0023	\$0.40	\$7.03	0.86
Honcho	0 %	1.2	3,045	\$1.37	\$0.0049	\$0.54	\$2.23	—
	50 %	2.6	5,681	\$2.42	\$0.0043	\$1.18	\$4.01	—
	90 %	12.2	39,083	\$15.59	\$0.0056	\$2.02	\$18.12	2.73
Graphiti	0 %	6.5	41,831	\$8.89	\$0.0318	\$0.24	\$9.24	—
	50 %	11.6	87,541	\$19.46	\$0.0345	\$0.27	\$19.84	—
Full context	0 %	0.0	0	\$0.00	\$0.0000	\$0.17	\$0.42	—
	50 %	0.0	0	\$0.00	\$0.0000	\$0.23	\$0.57	—
	90 %	0.1	0	\$0.00	\$0.0000	\$0.29	\$0.74	0.14

Table 12: Memory building and run cost per noise level. Total: build + answering at both budgets + judging. ¢/correct: total cost per correct 8K answer at 90 %. ^eEmbedding calls. Mem0’s 50 % cost is understated: a proxy restart left some of its calls without a run id.

Type	<i>n</i>	RAG	Hindsight	OpenViking	Mem0	Honcho	Full ctx.	No mem.
multi_hop	204	93.6	83.3	74.0	78.4	58.3	51.0	0.0
lookup	199	99.0	96.0	92.0	83.9	74.4	52.8	0.0
paraphrase	188	97.9	92.0	85.1	83.5	64.9	51.6	0.0
inference	115	95.7	87.0	69.6	81.7	66.1	55.7	0.0
temporal	108	99.1	93.5	89.8	79.6	74.1	44.4	0.0
perspective	105	99.0	94.3	82.9	83.8	66.7	60.0	0.0
rare_identifier	20	100.0	75.0	90.0	85.0	45.0	45.0	0.0
knowledge_update	20	90.0	80.0	80.0	65.0	50.0	60.0	0.0
preference	18	94.4	100.0	88.9	88.9	66.7	50.0	5.6
abstention	12	100.0	91.7	100.0	100.0	100.0	100.0	100.0
multi_session	11	72.7	45.5	63.6	45.5	45.5	9.1	0.0

Table 13: Accuracy (%) by question type at 8K, 90 % noise. Types with fewer than 25 questions are indicative only.

System	0 % noise	50 % noise	90 % noise
RAG	+1.4* [+0.6, +2.3]	+3.6* [+2.4, +4.9]	+5.1* [+3.6, +6.7]
Hindsight	+0.1 [−0.7, +0.9]	+0.2 [−0.7, +1.1]	+0.4 [−0.8, +1.7]
OpenViking	+0.0 [−0.6, +0.6]	+0.1 [−0.6, +0.8]	+1.5* [+0.5, +2.5]
Mem0	−0.1 [−1.1, +0.9]	−0.4 [−1.5, +0.7]	−0.6 [−1.6, +0.4]
Honcho	+19.8* [+17.3, +22.4]	+29.4* [+26.4, +32.3]	+30.4* [+27.6, +33.2]
Graphiti	−0.5 [−1.5, +0.5]	−0.1 [−1.0, +0.8]	—
Full context	+25.7* [+23.0, +28.5]	+42.9* [+39.7, +46.0]	+41.1* [+38.0, +44.0]
No memory	+0.1 [+0.0, +0.3]	+0.2 [−0.2, +0.6]	−0.2 [−0.5, +0.0]

Table 14: 8K minus 2K accuracy (points), paired.

System	Stage 1	Stage 1c	Change	Change by dataset			
				Enron	LoCoMo	LME	MSC
RAG	96.0	98.5	+2.5* [+1.5, +3.6]	+2.4	+4.4	+2.0	+1.2
Full context	95.8	98.0	+2.2* [+1.1, +3.3]	+2.8	+3.2	+2.4	+0.4
Hindsight	91.4	93.6	+2.2* [+0.3, +4.2]	+2.8	+2.8	+2.4	+0.8
Mem0	88.0	90.9	+2.9* [+0.6, +5.1]	+2.8	+7.2	-1.6	+3.2
Honcho	78.8	87.0	+8.2* [+5.1, +11.2]	+1.6	+2.4	-3.6	+32.4
OpenViking	71.8	84.3	+12.5* [+9.3, +15.5]	+40.4	+9.2	+1.2	-0.8
Graphiti	59.4	54.1	-5.3* [-8.6, -2.1]	+0.0	-8.8	+2.8	-15.2

Table 15: No noise, 8K. Stage 1: providers chosen by the router, reasoning off. Stage 1c (used throughout): one pinned provider, minimal reasoning. Same data and adapters, both judged by Jev; paired 95 % intervals. The baselines’ memory is unchanged, so their gain is the answer model’s own.