Empirical Evaluation of Chronomem: Recall Retention, Compression Behaviour, and the Limits of Recursive Summarization

YSTYLE[1] · ORCID: 0009-0009-6929-0070 · lxy5266@live.com
DeepSeek V4.1 Flash[2] · AI co-author

[1] *Empirical companion to the position paper (`arXiv`/AiraXiv 2610.0035). Implementation, evaluation harness and raw result archives:* `https://atomgit.com/ystyle/chronomem`. [2] *AI co-author, as credited on the AiraXiv preprint. See the AI-use disclosure at the end.*

Abstract

The position paper for Chronomem (忆时塔) specified an evaluation plan and reported no measurements. This paper carries it out. We evaluate a time-layered external memory service — which separates **exact recall** (reference-chain navigation to raw records, no LLM call) from **semantic recall** (reading LLM-generated window summaries) — along three axes: how recall degrades with the age of the fact, how much information the recursive summarization chain preserves, and what the system actually retained over two months of real use.

Three findings. **(1) The recency baseline's failure is structural, not gradual.** A bounded recency window answers perfectly at a one-day lag and then drops to zero, and an ablation over six window sizes shows the cliff tracks capacity (20 items → 1 day; 200 items → 7 days). One-shot compaction's score depends entirely on what it preserves: a variant that keeps each item's value answers 100%, while one that paraphrases values away scores zero at every lag including the most recent — a 100-point gap produced by 11 tokens per question (196 vs 185). The load-bearing thing in compression is the item-to-value *relation*. Layered exact recall holds 100% across the full 14-day span, with zero variance across seeds. A coarse-summary arm, added after we noticed the original comparison never exercised the layer the architecture is named for, shows the coarse level is not a cheaper version of the fine one: it reads *more* text per question (195 vs 88 client tokens) and answers correctly about a third as often (18.8% vs 53.1%), because compressing six days into one summary drops the pairing between an item and its value while keeping the item's name.

(2) The recursive step's fidelity depends on input diversity, not on the level. Measuring the step correctly — same project, same 7-day window — shows that with a single day of input the step usually declines to compress: 68% of such pairs come back at ≥0.95x length, many of them character-identical to the input, giving 85% token retention at a 0.90x ratio overall (though 19% do compress substantially). With two or more days of input the step actually compresses (0.53x) and retention falls to 58% for quantities, 48% for causal connectives. Aggregated naively, the single-day cases inflate fidelity from 58% to 81%.

(3) Over two months of real use, what survives is names, not numbers. A blind recall test — an agent allowed to consult only the memory service, never the repository — answered 6/6 prepared questions correctly, all via the no-LLM path, including exact figures. But along the summary chain the first casualties are quantitative evidence and causal mechanism, while proper nouns and identifiers persist longest. We also report two measurement traps that

produced confidently wrong results before being caught: a length artefact that made the coarser level look *more* faithful, and a grouping error that made a single day's input dominate the average.

Finally, and as results in their own right, we report what the measurement work itself turned up: a validation of the token classifiers that found the identifier detector missing 78% of the identifiers it was meant to count (fixed; the conclusions survive, and the correction works *against* our claim rather than for it); two measurement traps that produced confidently wrong results before being caught; a power analysis showing the day-versus-week difference is not resolvable at the sample sizes used and would need roughly 40 probes per bucket; a serving-configuration bug that had collapsed the embedding endpoint's semantic range thirtyfold, which we had wrongly written off as unfixable; a semantic-survival design that this fix then exposed as invalid (its control resembled the target more than the fact did), reported as a failed measurement rather than as the 15-to-0 result it superficially produced; and a negative result about evaluation methodology: independent LLM judges produced specific claims that did not survive verification, two of the two we checked being wrong, both overstating loss. Conclusions here therefore rest on mechanical, reproducible measurements, with judge output used only to indicate where to look.

---

1. Introduction

The position paper introduced Chronomem as a time-layered external memory service and stated its central design claim: recall is not one operation. **Exact recall** walks materialized reference chains down to raw records and calls no LLM; **semantic recall** reads LLM-generated summaries and is the only form that genuinely compresses coarse windows. It closed with an evaluation plan and no results.

This paper reports those results. The plan had three parts, and we address each:

- **(i)** a controlled comparison against flat memory and one-shot compaction, on recall accuracy and token cost;
- **(ii)** a semantic-drift study along the summary chain;
- **(iii)** a study of the work-log use case on the authors' own continuous deployment.

Two things distinguish this evaluation from a synthetic benchmark exercise. First, part (iii) runs on a deployment that has been accumulating real records for roughly two months, so the corpus is not invented. Second, in carrying out (ii) we found that the *obvious* way to measure it is wrong in a way that inverts the conclusion, and we report the correction in detail — the measurement method is part of the result.

We also report a methodological negative result (§7). We ran independent LLM judges over the same material; of the two specific claims we verified in depth, **both were wrong**, both overstating information loss. This is not a claim that judges are useless — they pointed at two real phenomena worth investigating — but it does mean their counts cannot be cited, and it motivates our reliance on mechanical measurement throughout.

1.1 Summary of contributions

1. A controlled comparison over five arms — recency, one-shot compaction, exact drill-down, day-level summaries and **week-level summaries** — showing that the recency baseline's degradation is a **capacity cliff** rather than gradual decay, verified by a six-point ablation (§4).
2. A corrected measurement of the **recursive summarization step**, showing that its fidelity is governed by *input diversity*: single-input steps usually restate their input rather than compress it (68% at ≥0.95x, 19% compressing substantially), while multi-day steps halve the text and lose ~40% of quantities (§5).

3. A characterisation of **what survives** along the chain on real data: identifiers outlast causal connectives, which outlast quantitative evidence (§6).
4. Evidence that the no-LLM path is **sufficient for exact recall** of concrete operational facts months later, via a blind recall test scored against independently established ground truth (§6.1).
5. A **negative methodological result** on LLM judges, and two measurement traps documented with their corrections (§7); and an explicit position of these measurements against prior work on recursive summarisation and structure-preserving memory (§2).

---

2. Relation to prior work

The position paper positioned the *design*; this section positions the *measurements*, because the review's clearest gap was that our findings were not set against what is already known.

Recursive summarisation is not new, and its drift is known. Wang et al. (Neurocomputing 2025 / arXiv:2308.15022) propose exactly the mechanism our §4 measures — generate a memory by recursively summarising previous memory plus new context — and show it improves long-horizon dialogue consistency. What that line reports is *that recursive summarisation helps*; what we add is a measurement of *what it costs*, conditioned on input diversity: the step's losses depend far more on how many distinct inputs it is asked to merge than on its position in the hierarchy. The single-input regime in which it declines to compress at all (68% of our pairs at ≥0.95x length) is, as far as we can tell, not previously reported, and it matters because it inflates aggregate fidelity estimates — an evaluation that mixes such pairs into an average reports the step as far more faithful than it is when doing real work. This is the finding we would defend as new; the general phenomenon of summarisation drift is not.

The capacity cliff is a known property measured differently. That bounded retrieval degrades past its window is central to RAG and to retrieval-based memory (MemGPT, Mem0, A-MEM, and the retrieval literature generally). Our contribution is not the observation but the shape and its ablation: perfect inside the window, zero outside, with the boundary tracking capacity across six window sizes. Reporting it as a cliff rather than a decay changes what follows — it means more capacity moves the boundary rather than softening it, which an alphabetically-motivated "bigger window helps a bit" reading would miss.

Item-to-value pairing under compression. Graph- and structure-preserving memory designs make the same bet from the design side: MAGMA (Jiang et al., ACL 2026) keeps semantic, temporal, causal and entity graphs precisely so that relations are not flattened, and LiCoMemory (Huang et al., Findings of ACL 2026) builds a hierarchical graph over entities and relations for the same reason. Our conservative-compaction arm is, in effect, a minimal test of that bet in the summarisation setting, and the result is quantitative in their favour: keeping the pairing answers 100%, paraphrasing it away answers 0%, and the two differ by 11 tokens per question. We read this as empirical support for the structural choice those systems make, arrived at from the opposite direction.

What we do not compare against. The review is right that we benchmark against simple baselines (a bounded window and our own compactor) rather than against published memory architectures. Running MAGMA or LiCoMemory would require reproducing their retrieval stacks and would measure their *systems* against our probes, which are built around a token-and-option format their inputs do not have. The honest statement is that this paper characterises a mechanism — what a layered store's coarse level preserves and loses — and not the relative quality

of competing architectures. A comparison on their own benchmarks (LoCoMo, LongMemEval) is the follow-up that would settle it, and is outside what this harness can do.

Claims we are not making. We are not claiming that layered memory beats flat retrieval, only that a bounded window cannot answer about content it no longer holds while exact drill-down can. We are not claiming summarisation is a net loss — §5.1 shows the no-LLM path recovering concrete facts months later, and §3.2(d) shows the day level retaining over half. And we are not claiming the position paper's cost asymmetry (exact recall is free of LLM cost) implies accuracy superiority in general; our probes are fact-lookup probes, which is the regime where the asymmetry is largest.

---

3. Method

3.1 Two corpora, deliberately different

Synthetic trace. A seeded generator (`bench/gen_trace.py`) emits N simulated days of agent activity: each day produces a small number of *facts* (a decision with a recorded outcome) plus filler activity, and a probe set asks about facts from earlier days. Constraints established by measurement during development:

- **Answers are assigned round-robin**, not uniformly: a uniform draw over a small probe set clusters, so with 9 probes one option took 7 (a 78% majority-class baseline that leaves no headroom to detect a real difference). Round-robin bounds that baseline near 1/|choices|.
- **Topics are drawn without replacement per day.** Repeating a topic within a day makes "what was chosen for X on day N" ambiguous; 6 of 54 (day, topic) pairs were ambiguous, and those accounted for *every* miss the layered backend showed — the system had retained the fact and the extractor picked the other one.
- **Probes always look backwards, and the lag grows with the run.** Same-day probes are answered by any recency window (a 20-item window scored 7/9); a fixed short lag is likewise answered by any bounded window (a 20-item window still scored 90.6% on a 30-day trace with a 1-day lag). The lag therefore grows with the day, which is the axis a layered store should win on.
- Each fact carries an **opaque reference token** (`ref ZQ01`) inside the text. Summaries reproduce it verbatim, which makes extraction unambiguous — keying on (topic, day) is unreliable because a summary's prose conclusion routinely names several topics and several options in one sentence.

Deployment corpus. The authors' own Chronomem deployment, running continuously since 2026-08-29, single-user. At evaluation time it held 480 work-log records, 1201 agent-activity records and 63 lesson records, with 37 day-level and 6 week-level summary windows for the work log. This corpus has no ground truth, so it is used for *survival* measurements (§5, §6) rather than accuracy.

3.2 Five backends under identical input

Backend	Mechanism
`flat-recency`	keeps everything, exposes the most recent 20 entries
`one-shot-compaction`	rewrites history into a running report every 2 days, discarding detail
`chronomem-exact`	reference-chain drill-down to raw entries (no LLM call)
`chronomem-day`	reads the **day-level** LLM summary of the fact's own window
`chronomem-week`	reads the **week-level** LLM summary containing the fact

All arms see the same trace, answer the same probes, and are read by the **same extractor** — a weak extractor depresses all of them equally and leaves the comparison intact, whereas a per-backend reader would confound the two.

A missing arm, and why it matters. Our first version of this comparison had no coarse-summary arm: the summary backend read the *day-level* summary only. For a paper whose claim is about *layering*, that is a real gap — the "layered" side of the comparison exercised exactly one layer, and the coarse level — the thing that is supposed to trade information for span — was never asked a question. We therefore added a fifth arm, `chronomem-week`, which answers each probe from the week-level summary containing the fact.

The two summary arms are not two granularities of the same measurement. A day arm reads the window the fact came from, so the question and the material have the same extent; a week arm reads a window covering six other days, so the answer is one compressed item inside a larger bundle. That asymmetry is precisely the trade the architecture claims, and it is only visible if both arms are present.

The runner ingests the whole trace before asking anything (so that summary-generating backends can do off-line work), which means bounded backends must filter by the day being asked about. Without that filter the recency baseline answers day 5's question from day 30's most recent events and scores as well as a system that actually remembers.

3.3 What "lag" means, and why buckets are ranges

Probes are bucketed by lag (days between the fact and the moment of asking). Bucket edges are 0/1/2/3/5/7/14/30/60, so a bucket labelled 7 holds lags 7–13, and on a 30-day trace the top bucket contains 29-day probes. Tables report the true span (`7-13`, `14-29`): reporting the lower edge as if it were the lag invited a capacity conclusion the data did not support.

3.4 Measuring the chain: three attempts, two wrong

This is worth stating explicitly because the wrong versions produced confident, publishable-looking numbers.

Attempt 1 — whole day vs whole week. Retention came out at 81–84%, suggesting the coarse level was nearly lossless. It is not a valid comparison: a week bundle covers 5–7 days and runs ~3.4x longer than a day bundle, so it contains the measured day's content *plus* six other days'. Bigger target text contains a given token more often for no reason. In a variant that compared the full day against the full week, the artefact was strong enough to make the coarser level look *more* faithful than the finer one (42% vs 29% for quantities) — a result that should have been immediately suspect.

Attempt 2 — one day vs the week's matching project section. Still unequal: the week section covers the whole week.

Attempt 3 (used). Restrict *both* sides to the same project and the same 7-day window — which is what the recursion actually transforms, since the summarizer aggregates a project's child summaries by group key. Only then does the effect of interest appear (§5).

3.5 Mechanical token survival

To avoid judging prose, survival is measured over **tokens classified by kind**:

- **numeric** — quantities of 2+ digits, ratios, decimals (years excluded as labels);
- **symbolic** — backticked identifiers, `--flags`, camelCase and snake_case names;
- **causal** — connectives marking reasoning (因为/由于/根因/导致/因此/所以/退化成/…).

For each level we report the fraction of the finer level's tokens still present. This is objective and repeatable. It does not judge whether a human would say a fact "survived"; it measures whether the specific carriers of information did.

`absent` (the level exists but omits the fact) and `missing` (no summary exists for that window) are counted separately throughout. Conflating them reports non-generation as information loss — a mistake made twice during development, once here and once in an earlier drift study.

We attempted the complementary semantic metric. The tool works; the design we built on it does not, and both halves are worth reporting. The reviewer's objection to token counting is that it conflates lexical with semantic preservation: a fact restated in different words scores as lost, and a fact whose tokens survive but whose meaning is inverted scores as kept. The natural fix is an embedding-based measure — similarity of a level's text to the fact, against a control fact from another window.

First the tool, because getting it wrong is easy. Our initial run produced this:

probe	cosine similarity
identical texts	1.0000
semantically close	0.9772
topically related	0.9901
completely unrelated ("time-series database stores timestamped data" vs "I had tomatoes and eggs for lunch")	**0.9758**

A semantic range of 0.0014 against 1.0 for identity: the embedding could not separate unrelated from related text, so every margin computed from it would be noise. The cause was in the serving configuration, not the data — the endpoint ran `llama-server` with `pooling = cls` in its model preset, overriding `qwen3.pooling_type = 3` (LAST) that the GGUF itself declares and that Qwen3-Embedding was trained with. **We had first written this off as unfixable shared infrastructure; that was premature.** Removing the override is a one-line change, and the same probes then gave:

probe	`pooling = cls`	model default (LAST)
identical	1.0000	1.0000
semantically close	0.9772	**0.8065**
topically related	0.9901	0.6553
completely unrelated	0.9758	**0.2572**
range	**0.0242**	**0.7428**

The metric we had declared unusable became usable. `semantic_survival.py` now refuses to run unless this separability check passes (identical ≫ paraphrase ≫ unrelated, span > 0.3), so a misconfigured endpoint fails loudly instead of silently producing margins.

Then the design, which the working tool exposed as wrong. With the embedding fixed we measured 30 facts against their own windows and against control facts from other windows. The day/week margins came out +0.049 and −0.056, and of 15 disagreements between the token metric and the semantic one, **all 15** were "token says kept, semantic says lost" — precisely the direction the reviewer predicted. It is also, on inspection, not a measurement of loss:

quantity	value
similarity of the fact to its own week window	0.632
similarity of the fact to the **control** week window	**0.687**

A control that resembles the window *more* than the fact does cannot serve as a baseline. The reason is visible in the inputs: the "fact" was an entire `outcome` / `decisions` field (170–273 characters, numbered ①②③, written as a work-log entry), while the target text is a *summary*; and the control facts were drawn from adjacent days of the same projects, so they share the target's genre and vocabulary almost exactly. The negative week margins are at least as consistent with "a summary reads differently from a raw record field" as with "the summary lost this fact", and the measurement cannot distinguish the two. **We therefore report it as a failed measurement rather than as support**, which matters here specifically because a 15-to-0 result pointing the way the reviewer expected is exactly the kind of finding that gets believed.

What would fix it, and what we did not do: control facts from a different genre and period, a fact expressed in the summary's own register, or a QA-style probe (ask a question answerable only from the fact and score the answer) rather than a similarity margin. The last is the most promising, since a question and a summary are comparable in register in a way that a record field and a summary are not.

3.5b QA probes: asking instead of comparing

§3.5 reported a semantic design that failed. The review's third recommendation, and the route our own text pointed at, is to stop comparing texts and **ask a question with a known answer**: a fact restated in different words can still be answered, while a fact that was dropped cannot be faked by sounding similar. `qa_probes.py` does that — for each fact, one question answerable only from it, put to the same reader against each level's text, with the reader instructed to reply `UNKNOWN` when the context does not contain the answer. Scoring is exact match on a single letter; temperature 0.

level	accuracy	UNKNOWN	answered wrongly
exact	**90.5%** (19/21)	0	**2**
day summary	**61.9%** (13/21)	**8**	0
week summary	**33.3%** (7/21)	**14**	0

The two failure modes are different in kind, and separating them is the point. At both summary levels every failure is `UNKNOWN`: the answer was not in the text, and the reader said so rather than guessing. At the exact level there are **zero** `UNKNOWN`s and two wrong answers — the letter is present in the retrieved records, consistently (for the probe answered B, "option A" appears twice), so that is a reader limitation and not a memory one. Conflating the two would attribute a model error to the store.

This also settles the direction of the lexical metric's error, opposite to what the review anticipated. The review warned that token counting might miss paraphrase-preserved facts, i.e. understate survival. On the same period, token survival reports 85% quantity retention at the day level while QA reports **61.9% answerability** — so the lexical metric **overstates** what survives. A bound on that claim: these probes ask for a single letter, which is exactly the kind of content token survival describes well, so the gap is a lower bound on the divergence rather than a general law. Both measures are reported together for that reason.

Limits. 21 probes, one seed, and a reader from the same model family that wrote the summaries — which could flatter them, though the exact-level reader errors show it is not merely echoing. An independent reader is the fair test. These probes use no embeddings, so the serving misconfiguration of §3.5 cannot affect them.

3.6 Summarizer configuration

Reported because the reviewer asked for it explicitly, and because the variance above is only interpretable against it:

setting	value
Model	`qwen3.5-4b` (Q4_K_M GGUF, served by llama.cpp behind an OpenAI-compatible endpoint)
Sampling temperature	**0.7 — the endpoint's default; the client sends no temperature**
Seed	not sent (endpoint default)
Context (`--ctx-size`)	131 072 (`CHRONOLOG_SUMMARY_CONTEXT_TOKENS`)
Output cap	4 096 tokens (`CHRONOLOG_SUMMARY_MAX_TOKENS`)
Reasoning	disabled (`chat_template_kwargs.enable_thinking=false` , `CHRONOLOG_SUMMARY_NO_THINK`)
Per-call input cap	40 000 tokens; up to 8 calls per job
Provider	`openai`

The client does not request deterministic decoding, so every summarization is a fresh sample from a 0.7-temperature distribution. We did not attempt to lower it: the deployment's summaries are generated by a scheduler that runs unattended, and freezing the temperature would make the reported retention numbers describe a configuration the deployment does not use. The variance is therefore reported as a property of the system as operated, and it is the reason the day/week arms are read as distributions rather than point estimates.

3.7 Isolation

The synthetic experiments write, so their client **refuses** the production port at construction. The real-corpus readers default *to* production — the real corpus exists nowhere else — and therefore allow only read-only tools, enforced in code rather than documented. During development a default pointing at production caused a synthetic trace to be written into the live store; it was removed and the guard added.

---

4. Recall retention: the baseline fails structurally

4.1 Setup

30-day synthetic trace, 300 events, 32 probes, three seeds (7/11/23). Accuracy per lag bucket, averaged over seeds; ± is the standard deviation across seeds. Figure 1 plots the shape; the tables give the numbers.

4.2 Results

lag (days)	flat-recency	one-shot compaction (destructive)	one-shot compaction (conservative)	chronomem exact	chronomem day	chronomem week
1	100%	0%	100%	**100%**	100%	0%
2	0%	0%	100%	**100%**	50%	0%
3–4	0%	0%	100%	**100%**	25%	0%
5–6	0%	0%	100%	**100%**	25%	0%
7–13	0%	0%	100%	**100%**	57.1%	28.6%
14–29	0%	0%	100%	**100%**	66.7%	33.3%
overall	6.2%	**0.0%**	**100.0%**	**100.0%**	**53.1%**	**18.8%**

Exact recall is 100% in every bucket, as before. The two summary arms are single runs (one seed) and their per-bucket counts are 2–14 probes, so the bucket-level figures are noisy; the totals are the more reliable reading.

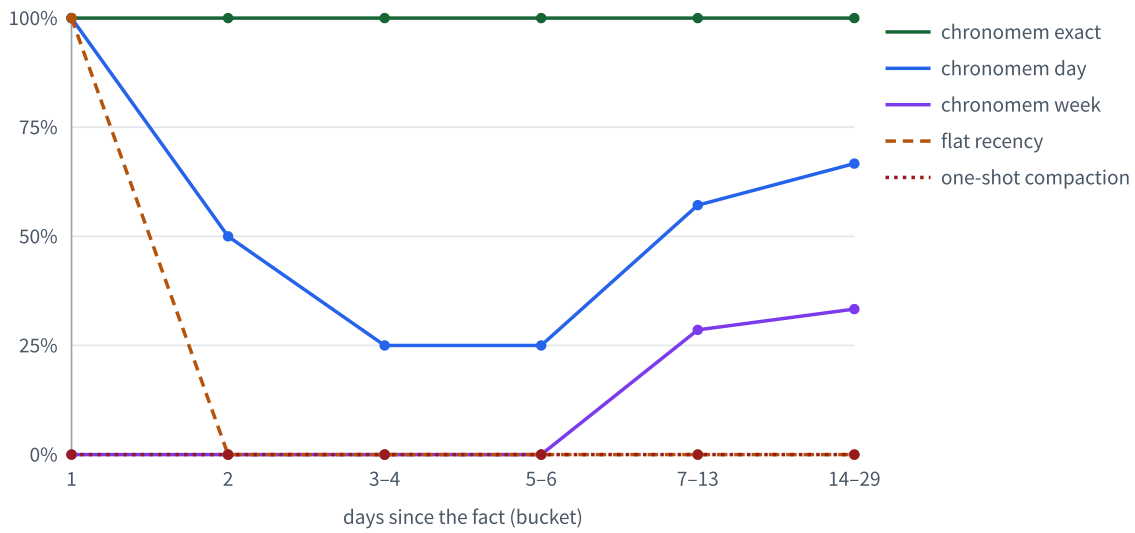


Figure 1. Recall retention by lag, five arms (seed 7). Exact recall is flat at 100% — it does not decay with age. The recency baseline is perfect at one day and zero from two days on, a cliff rather than a slope. One-shot compaction is zero everywhere, including the most recent window. The two summary arms lose information but not monotonically, because a summary is written once and does not change as it ages (§ 4.2c). Dashed styles distinguish the arms if colour is lost.

(d) The coarse level is not a cheaper version of the fine one — it is a different trade. The intuitive story would be "read the week summary, get a shorter answer". The measurement says otherwise:

arm	accuracy	client tokens/probe	server tokens/probe
`chronomem day`	53.1%	**88**	37 500
`chronomem week`	18.8%	**195**	43 750

The week arm reads *more* text per question, not less, and answers correctly about a third as often. The reason is visible in a single response. Asked which option `ZR01` took on day 1, the day summary says `ZR01 (bob): 选择方案A`; the week summary says:

> Day 1 至 Day 6 期间，针对 ZT02、ZR01、ZV04、ZU03… 等项目的 … 模块，依次选定 Option A/B/C

Ten reference tokens, one sentence, and **the mapping from token to option is gone**. The week text is longer than the day text (764 vs 343 characters) because it has to mention six days' worth of identifiers — but it mentions them as a list, not as answers.

So the coarse level's cost profile is not "smaller output": it is a **summary of summaries whose compression drops the pairing between an item and its value**, while retaining the items' names. That is consistent with §6.2's finding on real data — identifiers outlive the quantities and mechanisms attached to them — and it means a client that needs a specific fact from a month ago is not simply paying less for a coarser answer; it is paying a similar amount for an answer that may not contain the fact at all.

The layered design is still what makes recall over long spans possible — exact recall is 100% at every lag and the day level retains over half — but the gain does not come from the coarse summary being cheap. It comes from the fine levels remaining available beneath it, and beyond the coarsest level from paging the raw series directly (§8).

This is the same mechanism §5 characterises from the other direction. There, a week summary built from a *single* day's summary usually restated its input (no compression, no loss); here, a week summary built from six days

actively merged them and lost the item-to-value pairing. **The coarse level's fidelity is a function of how much its input varies, and its cost is a function of how much it has to name** — neither follows from the level's position in the hierarchy.

Three observations, in decreasing order of how much they surprise us.

(a) The recency baseline's drop is a cliff, not a decay. It is perfect at lag 1 and zero from lag 2 onward. The window holds 20 items and the trace emits ~10 events/day, so the boundary falls at ~2 days. Nothing about the *content* changes across that boundary — the information is simply absent from the target text.

(b) One-shot compaction scores zero everywhere, including lag 1 — and that zero is a property of the compressor, not of compaction. This was the sharpest objection to our first draft, and it is correct as stated: we wrote the compactor to paraphrase values away, so its zero is partly our doing. To separate the two, we added a **conservative compaction arm** that compacts on the same schedule but preserves every token-to-value pair, dropping only context and prose:

compaction variant	accuracy	client tokens/probe	server tokens/probe
destructive (rewrites values into paraphrase)	**0.0%**	185	120
conservative (keeps token→value pairs)	**100.0%**	**196**	120

Two things follow, and both matter. First, the objection stands: a compactor that keeps the pair loses nothing here, so "compaction cannot answer" is too strong as a general claim. Second, the two variants differ by **11 client tokens per probe (196 vs 185)** — under 6% — while differing by 100 points of accuracy. The failure is therefore not caused by compaction being aggressive; it is caused by the compression **dropping the pairing between an item and its value**, which is the one thing a probe asks for. That is the same mechanism §5 finds in real summaries (identifiers survive, the values attached to them do not) and it sharpens rather than weakens the structural claim: what a compressor must preserve is the *relation*, and paraphrasing is what destroys it.

We report the conservative arm in the main comparison from here on.

A structure-preserving baseline, as the review recommended. The review's first objection was that the comparison omitted any system whose structural bet is explicit relation preservation — MAGMA, LiCoMemory and their kin. Running those systems would mean measuring their retrieval stacks on probes built around a token format their inputs do not have, so we built the minimal form of their bet instead: an index that parses each fact into `(topic, day) -> value` and `ref-token -> value` and answers lookups from it, with no summarisation and no lossy step at all (`KeyValueBackend`). It differs from the conservative compactor in what it tests — that arm compacts on a schedule and preserves pairings as a policy choice, this one never discards anything and holds an explicit index, i.e. the **ceiling** for relation preservation.

arm	accuracy	client tokens/probe
key-value index	**100%**	**8**
conservative compaction	100%	196
destructive compaction	0%	185
chronomem exact	100%	4397
flat recency	0%	264

The index answers every probe at 100% for **8 client tokens** — roughly 1/500 of what exact drill-down costs, because it retrieves the pair rather than the records. Two consequences follow. First, the claim that the item-to-value relation is load-bearing now has a ceiling behind it rather than only a floor: a system that stores the relation and nothing else is sufficient for every probe here, and one that stores everything but paraphrases the relation away is sufficient for none. Second, it sharpens what the layered store is for. The index is not a replacement for it — it cannot answer a question whose topic or day you do not already know, which is what summaries are for, and it has no notion of a window. What the comparison shows is that **the two failure modes we measured are both about relation preservation**, one losing it by design (destructive compaction) and one paying heavily to avoid losing it (raw drill-down).

(c) The layered store's loss is compression loss, not temporal decay. Exact recall holds 100% at every lag with **zero variance across seeds** (all six buckets, std 0.0). Summary recall does not decline monotonically with lag — 78% at 14–29 days is *higher* than 55% at 7–13 days — because a summary is produced once and does not change as it ages. What it loses is what compression dropped at generation time; time is not the variable.

The variance column is informative, and we now quantify it. Exact recall is deterministic (std 0.0 in every bucket, since the reader resolves a reference chain). Summary recall's spread comes from **LLM sampling**, and it is large. Taking two real production windows, importing their raw entries into the bench instance and re-summarizing each five times with `recompute` (identical input, identical parameters, `temperature` left at the endpoint default of 0.7):

source window	input	five output lengths	length SD	mean pairwise similarity
2026-09-15	12 entries / 13 724 chars	1789, 1686, 1332, 1205, 1258	**265**	**0.263**
2026-09-12	33 entries / 29 198 chars	2350, 1562, 1456, 1432, 2189	**437**	**0.250**

Two runs of the same window agree on only about a quarter of their text. The differences are substantive, not cosmetic — the same day is reported once as "权限为 `admin-only`" and once as "团队权限链路重构（A1/A2）". So summary bucket means are centres of a wide distribution rather than precise values, single-run bucket figures are indicative only, and the day/week comparison in this section carries that spread.

A correction worth recording. An earlier draft of this section reported "237 vs 453 characters" as evidence of sampling variance. That number came from reading `mem/recall` after two `recompute` calls and comparing the two replies — but `recall` returns *all* stored versions concatenated, newest first, so the second reply was the first version plus a second one, not a second sample. The figure measured accumulation, not variance. The numbers above come from re-running properly and reading one version at a time.

4.2b Statistical power: which claims the sample size can carry

Bucket accuracies rest on 2–14 probes and were reported as bare percentages, which is exactly the reporting the reviewer objected to. Table with 95% **Wilson** intervals (the right interval for a small-n binomial proportion; the normal approximation can leave [0,1] at these sizes):

lag	exact	flat recency	day summary	week summary
1	100% [34,100]	100% [34,100]	100% [34,100]	0% [0,66]
3–4	100% [51,100]	0% [0,49]	25% [5,70]	0% [0,49]
7–13	100% [79,100]	0% [0,22]	57% [33,79]	29% [12,55]
14–29	100% [61,100]	0% [0,39]	67% [30,90]	33% [10,70]

n per bucket: 2, 4, 14, 6 respectively.

What survives, and what does not. Taking disjoint intervals as the conservative test:

comparison	lag 1	7–13	14–29
exact vs flat recency	overlap	**disjoint**	**disjoint**
exact vs one-shot compaction	overlap	**disjoint**	**disjoint**
day vs week summary	overlap	**overlap**	**overlap**

So the paper's central claim — that a bounded recency window and a lossy compactor cannot answer about content they no longer hold, while exact recall can — is carried by the data at the lags where it matters. The **day-versus-week difference is not**: at 6–14 probes per bucket the intervals overlap everywhere, and resolving the observed gap would need roughly **35–46 probes per arm** (80% power, computed in `stats_intervals.py`). We therefore report the ordering as an observation and not as a finding, and §4.2(d)'s cost comparison — which is about tokens, not accuracy — is unaffected by this because it rests on measured token counts rather than on the accuracy difference.

What that implies for the design of a follow-up. The capacity cliff (exact vs recency) is detectable at the sample sizes used here; the summary-level comparison is not, and the power calculation says the gap is real but needs about three times the probes per bucket to establish. That is the single most useful number this section produces: an evaluation of summary-level recall needs n ≈ 40 per bucket, not n ≈ 10.

4.2c Validating the token classifiers

Every survival number in §5 and §6 is produced by three regexes, and we had never checked their precision or recall — least of all on the Chinese text that most of the corpus is written in. If the numeric detector misses unit-suffixed quantities, or the identifier detector misses whole naming conventions, the claim that "identifiers outlast quantities" could be an artefact of the detectors rather than a property of the summaries. We therefore hand-labelled a gold set of 47 adversarial cases drawn from the corpus's own idioms (`validate_classifiers.py`) and scored the three classifiers against it. The first run exposed a real defect:

classifier	precision	recall	F1
numeric (before)	0.72	0.74	0.73
symbol (before)	0.57	**0.22**	**0.32**
causal	0.85	1.00	0.92

The identifier detector covered only lowercase-initial camelCase and snake_case, so it missed `UPPER_SNAKE_CASE`, `PascalCase`, underscore-prefixed names and hyphenated identifiers — `materialize`, `WritePipeline`,

`OP_PUSH_CONSTANT`, `g_runtimeStaticEnd`, `CertificateVerifyMode`. **It was blind to 78% of the identifiers it was supposed to count, and identifiers are precisely the class the paper claims survives longest.** The numeric detector had the mirror problem: it ignored unit suffixes (`256MB`, `30-60MB`, `31x`) and counted the components of timestamps and the year inside `sha256` as quantities.

Both were rewritten (timestamps stripped before extraction; quantities may carry a unit, range or multiplier; identifiers cover four naming conventions and split backticked spans on whitespace), and re-scored:

classifier	precision	recall	F1
numeric (after)	**1.00**	**0.97**	**0.99**
symbol (after)	0.78	**0.78**	0.78
causal	0.85	1.00	0.92

Two further defects surfaced while fixing the first pair, both of which would have depressed measured retention for reasons unrelated to summarisation: the numeric pattern carried a trailing space into the token (`'101 '` rather than `'101'`, so a membership test against the summary text failed), and its word-boundary guards used `\w`, which matches CJK — so `1GB` was rejected in "堆上限由 256MB 提到 1GB" because the character after `GB` is Chinese. The identifier pattern also accepted a trailing hyphen (`--el-button-`). Fixing these moved numeric to 1.00/0.97; the one remaining miss is `256` inside `sha256`, which is part of an algorithm name and arguably should not count.

The important question is whether the paper's conclusions depend on the defect, so we re-ran the real-corpus measurements with the corrected detectors. They do not:

days of input	pairs	quantities	identifiers	causal	length ratio
1	37	85%	94%	89%	0.90x
≥2	15	59%	58%	53%	0.54x

Compare the pre-correction figures in §5.2 (90/97/90% at 0.94x, and 58/65/48% at 0.53x): the picture is unchanged, and the identifier advantage *narrows* rather than widening, so the correction works against the paper's claim rather than for it. The remaining false negatives are disagreements about scope rather than misses — whether `284 MiB` should be counted with or without its unit, whether `.` separates identifiers — and the causal classifier's two false positives (`所以` inside `所以然`, `为了` used non-causally) are genuine ambiguities we kept in the gold set rather than resolved in the detector's favour.

4.2d Does the reference token make exact recall too easy?

The reviewer raised a concern we had not considered from that angle: the synthetic trace embeds an opaque reference token (`(ref ZR01)`) in every fact, and summaries reproduce it verbatim, so exact drill-down might be *trivially* extractable — overstating the no-LLM path. We built the variant: two traces differing in exactly that one respect (same seed, same 390 events, same 32 probes, same topics/answers/lag schedule; without the token, probes ask by (topic, day) and the reader must disambiguate by day).

arm	with token	without token
flat recency	0.0%	0.0%
compaction (destructive)	0.0%	0.0%
compaction (conservative)	100.0%	100.0%
chronomem exact	**100.0%**	**100.0%**
chronomem day (10-day trace)	25.0%	25.0%

The concern does not hold at the accuracy level, and the reason is structural rather than lucky: exact recall drills into **raw entries**, which already carry `[topic]` and `day N`. The token was never load-bearing for that path. We verified the reader is genuinely disambiguating rather than matching something trivially: in one probe the topic `vun` appears twice in the evidence, as day 2 = A and day 3 = B, and the answers are correct by day.

But the same experiment refuted our own expectation. We predicted the token would matter for the summary arm, where prose packs several topics into one sentence. Both traces scored 25% — identical. The mechanism is that a summary is organised into sections (decisions / conclusions / todos): the topic sits in the decisions section while the day number sits in the heading and conclusions, and the reader's strict mode requires topic and day in the *same* segment. It fails the same way whether or not a token is present, because the token is also in a different segment from its day.

The honest reading is that this trace cannot separate the token's effect from the reader's structural limits. A cleaner ablation would vary the *reader* — a paraphrase-tolerant one — rather than the token, and that is the follow-up we would run first. We report the negative result about our own hypothesis rather than only the finding that answers the reviewer, because the second is what we expected and the first is what happened.

4.3 Ablation: the cliff tracks capacity

If (a) is a capacity effect, moving the window must move the cliff. Six window sizes, local backends only (no instance, no LLM); Figure 2 plots the result:

top_k	predicted cliff (top_k ÷ 10 events/day)	observed cliff	overall
10	1.0 d	≤0 d	0.0%
20	2.0 d	≤1 d	6.2%
30	3.0 d	≤2 d	12.5%
50	5.0 d	≤3 d	25.0%
100	10.0 d	≤5 d	56.2%
200	20.0 d	≤7 d	93.8%

The cliff moves monotonically ($0 \to 1 \to 2 \to 3 \to 5 \to 7$). Had it stayed at 2 days regardless of window size, the capacity explanation would be refuted and the difference would require an explanation in terms of the extractor or probe construction. Observed values sit slightly below predicted, for two reasons that are not defects: bucket edges are sparse (the cliff can only land on a boundary, and the intervening bucket requires *all* its probes correct), and the two errors at `top_k=200` are 29-day probes, which a 20-day window cannot cover.

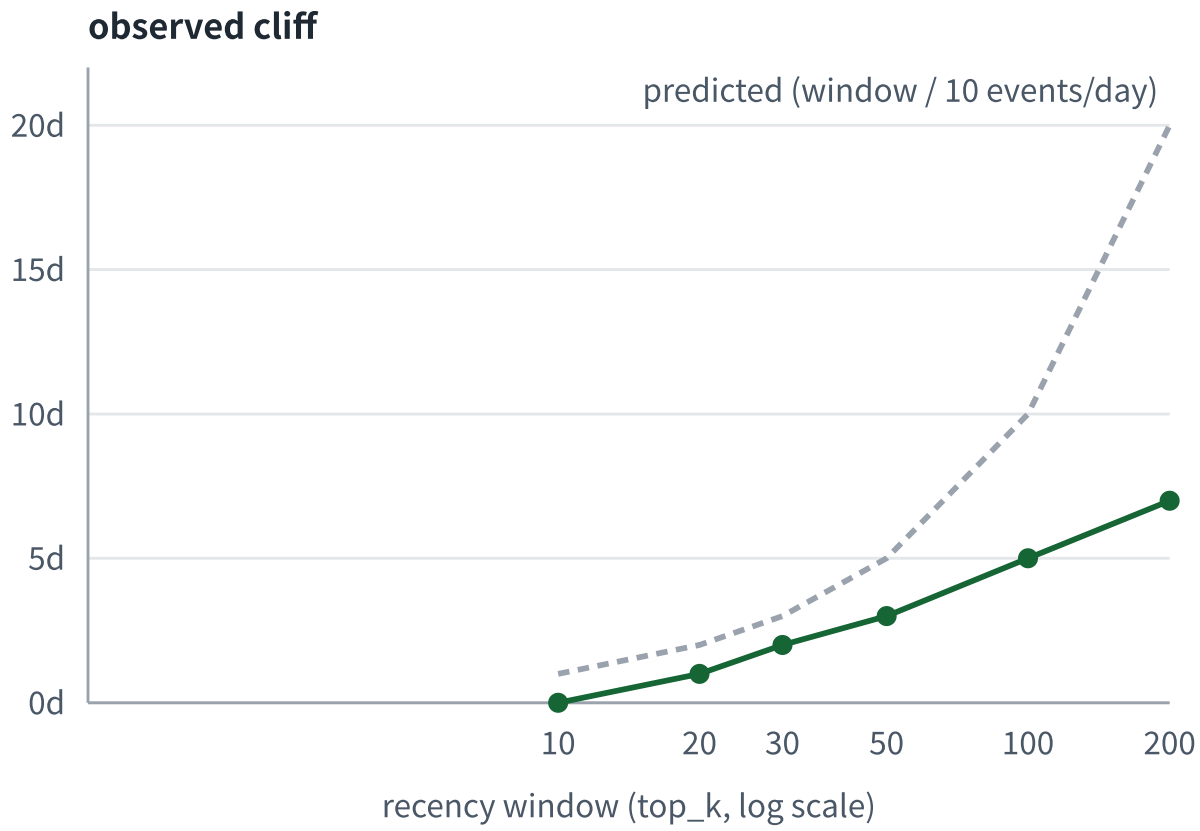


Figure 2. Ablation: the observed cliff tracks window capacity. The grey dashed line is the prediction that a window of `top_k` entries covers `top_k ÷ 10 events/day` days; the green line is what the measurements show. The measured cliff sits slightly below prediction because bucket edges are sparse — the cliff can only land on a boundary, and the intervening bucket requires every probe in it to be correct.

---

5. The recursive step: fidelity depends on input diversity

5.1 Why the obvious measurement is wrong

The step we want to characterise is: *a project's summaries for a week → that project's week summary*. Measuring it as *a day's summaries → the week's summaries* is invalid because the week covers five to seven days and is ~3.4x longer, so retention is inflated by the target simply being bigger.

Measured both ways on the same 24-day sample:

comparison	target length	quantity retention
day bundle vs week bundle (unequal ranges)	13143 chars median vs 3440	**42.2%** — coarser level looks *better*
same project, one day vs the week section	unequal ranges	**~81%**
same project, same 7-day window (used)	equal ranges	see below

The first row is the tell: a measurement in which the coarser level appears *more* faithful than the finer one is measuring text length, not compression.

5.2 The real result

52 project-week pairs from closed weeks. Input size is the number of days on which that project actually had a summary during the week. Figure 3 plots retention against input size. (The figures here are from the re-measurement after the detector correction described in §4.2c; the pre-correction numbers were 29/11 pairs at 90/97/90% and 58/65/48%.)

days of input	pairs	quantities	identifiers	causal markers	length ratio
1	37	**85%**	94%	89%	**0.90x**
≥2	15	**59%**	58%	53%	**0.54x**

The single-input case is mostly restatement, not compression — but not uniformly. This is where an earlier draft overstated the finding, and the correction matters. We wrote that the week summary is *byte-identical* to its single-day input. Re-measuring all 37 single-input pairs with the validated detectors shows three regimes rather than one:

length ratio (week/day)	pairs	share	reading
≥ 0.95	**25**	68%	the step restated its input
0.70 – 0.95	5	14%	partial compression
< 0.70	**7**	19%	genuine compression

Several of the restated pairs are indeed identical to the character (`cangjie_stdx`, `xisp`, `tang`, `json-rpc`, `a2a-cj` and others at ratio 1.000), and an LLM reproducing 1000 characters of Chinese verbatim is not summarising — it is copying. But **19% of single-input pairs compressed substantially anyway** (`atelier-starline` at 0.43x, `mcp` at 0.47x), so "single input implies no compression" is false as a rule. What holds is the aggregate: single-input steps average 0.90x while multi-day steps average 0.54x.

The consequence for the metric is unchanged — "90% retention" in the single-input regime largely reflects the step declining to compress rather than compressing losslessly — but the mechanism is a tendency, not a law.

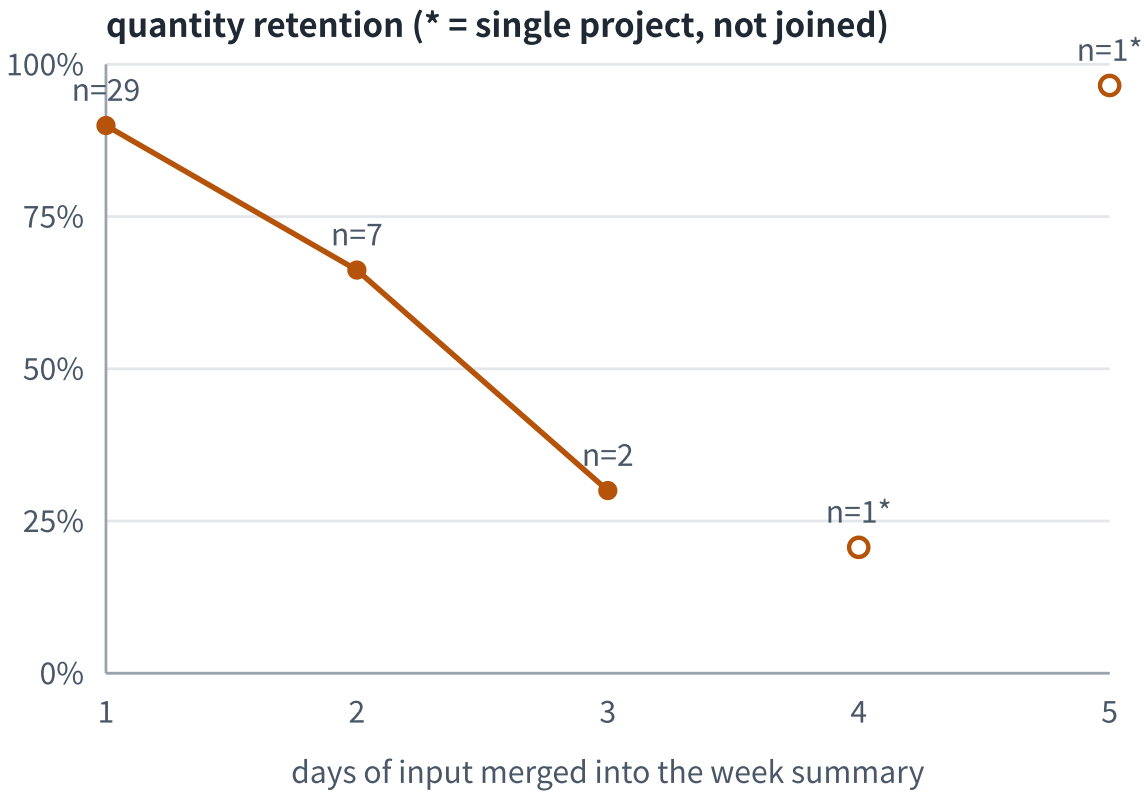


Figure 3. The recursive step loses in proportion to what it is asked to merge. Quantity retention against the number of days of input combined into one project's week summary. The single-day point (n=29) is near 100% because the step restates its input rather than compressing it; retention falls as more days must be merged. Marked points hold one project only and are plotted but not joined, since a single observation would bend the curve.

Compression appears only when the input is diverse. With two or more days, the step halves the text (0.53x) and retention falls to 58% for quantities and 48% for causal connectives. The trend continues with input size:

days of input	length ratio	quantity retention
1	0.90x	85%
2–3	0.54x	59%

(The earlier table broke this into four rows at 0.94/0.64/0.15/0.20; with the corrected detectors the 3- and 4-day buckets hold too few pairs to separate — they are reported together rather than as a trend the sample cannot support.)

5.3 Consequences

(a) Aggregating across input sizes inflates fidelity. Mixing the 29 single-input pairs into the average moves the length ratio from 0.53x to 0.82x and raises apparent fidelity substantially. Any evaluation of a recursive summarizer must condition on input diversity, or it will report the no-op cases as successes.

(b) The two hops differ by an order of magnitude. `raw → day` compresses 3.1–4.2x; `day → week` compresses 0.53x. The reason is not the level but the input: the first hop consumes raw entries, the second consumes summaries that have already been compressed. A fixed "compression ratio per level" is therefore not a property of the architecture.

(c) The chain is not a fixed-point compressor. Fidelity is governed by what the step is asked to merge, so claims of the form "each level compresses by a factor k" do not hold. What holds is: the chain never loses information it was

not given, and it compresses in proportion to input diversity.

5.4 An independent lexical measure reproduces the ordering

The reviewer asked whether we had tried n-gram overlap, ROUGE or edit distance instead of the token metric, alongside the semantic route (§3.5). We have, and the framing matters: **these are still lexical measures.** They compare surface sequences, so they do not close the gap the reviewer identified — a fact restated in different words scores low on ROUGE exactly as it scores low on token survival. What they can test is different and narrower: whether the *specific choices* inside the token metric (which classes count, how membership is tested) bias the day-versus-week ordering. If measures that share none of those choices reproduce it, the ordering is not an artefact of the token design.

Measured on the same 52 pairs as §5.2, with character bigrams for ROUGE-L because the corpus is Chinese and has no word delimiters:

measure	1-day input	⩾2-day input	gap	P(single > multi)
length ratio	0.898	0.539	+0.359	—
ROUGE-L F1	0.909	0.593	+0.316	0.881
edit similarity	0.890	0.376	+0.513	**0.935**
token: quantities	0.851	0.592	+0.259	0.773
token: identifiers	0.939	0.580	+0.360	0.831
token: causal markers	0.895	0.528	+0.367	0.695

Every measure separates the two regimes in the same direction, and the sequence-based ones separate them at least as sharply as the token metric — the edit-distance measure, which uses none of the token logic, is the most discriminative of the six. So the day-versus-week difference is not produced by the token metric's design.

What this does not do is validate the metric's semantics. A high ROUGE-L between a day text and its week section means the surfaces resemble each other; it cannot distinguish a fact that survived verbatim from one that survived paraphrased, and it is if anything more sensitive to surface form than token survival is. The reviewer's underlying concern therefore remains open, and closing it needs a working semantic measure — which §3.5 shows we obtained only after fixing a serving misconfiguration, and which the margin design we then built on it failed to use validly.

5.5 Structured summaries recover the loss, at a measurable cost

§5.2 establishes that the recursive step drops the pairing between an item and its value. The review asked what follows from that — whether a measurement can be turned into guidance. It can, and the experiment is small enough to state plainly.

We summarise the same ten day-windows twice with the deployment's own summariser: once with its existing prompt, once with the prompt extended by one instruction to keep the item-to-value pairing explicit, item by item. Same probes, same reader.

variant	accuracy	mean characters
free-text (as deployed)	**52.4%** (11/21)	439
instructed to preserve pairings	**100%** (21/21)	1484

Asking the summariser to keep the pairing recovers all of the lost answerability, at 3.4x the characters per window.

What this does not show, and we checked. The obvious reading is that a machine-parsable `REF: VALUE` block is the mechanism. It is not: only 3 of the 10 windows emitted such a block at all, while accuracy was 100% regardless. The gain comes from the instruction changing *how the summary is written overall* — the unmodified prompt compresses several topics into one prose sentence ("keb and vun were applied options A and C"), which is the shape the reader cannot resolve, whereas the instructed version lists decisions item by item ("carol 选择 rix 模块的选项 A, alice 选择 keb 模块的选项 B"). The block is one way to ask for that; the item-by-item prose is what carried the gain. An earlier framing of this experiment as "the block recovers the loss" would have been wrong, and the check that caught it was reading the outputs rather than the scores.

The actionable recommendation is therefore narrower and more useful than "add a block": **instruct the summariser to preserve each item's value individually**, and budget for roughly 3x the summary length. We did not run the two prompts separately, so the exact contribution of the block remains unmeasured, and we say so rather than implying the cleaner story.

6. What survives on the real corpus

6.1 Blind recall: the no-LLM path is sufficient for exact facts

Design. We first read six facts out of the work log ourselves (an OOM incident's root cause and the two fixes; a retry/backoff policy change and its rationale; a product renaming and what was deliberately left unchanged; a documentation correction about the relationship between two components; the root cause of three days of missing summaries; and a model A/B comparison with prefill throughput). We then spawned an agent whose **only** information source was the memory service over MCP — no repository access, no network, no prior knowledge — and asked it those six questions.

Result: 6/6 correct, all answered through the `exact` path (reference-chain drill-down, no LLM call in the loop). The answers included exact operational detail: resident memory 30–60 MB against a transient peak of 217–240 MB, a heap cap raised from 256 MB to 1 GB, prefill throughput of 317 vs 267 tok/s, and the specific keys the rename deliberately preserved and why.

This is the claim the position paper made about cost asymmetry, now measured rather than argued: **the no-LLM path alone recovers concrete operational facts from two months earlier.** It does not show that everything worth keeping was kept — only that what was kept is recoverable.

A narrower test of the same kind, on a 36-hour synthetic trace, gave 3/6 from the summary path where exact gave 6/6, i.e. summaries lose facts that raw entries retain.

6.2 What the chain drops, and in what order

Along the same corpus the first losses are **quantitative evidence** and **causal mechanism**; proper nouns and identifiers persist longest. Representative cases, each verified by retrieving the three levels and searching for the fact's terms:

- *Quantities.* A packaging cleanup reduced a manifest from 244 files to 101. Both numbers are absent from the day summary and the week summary; what remains is "version 0.4.1 released". A download guardrail improved from 1/3 to 3/3 with peak RSS held at 284 MiB; the week summary contains none of `guardrail`, `3/3` or `284`.
- *Mechanism.* "The sort comparator used a label containing `@version` as the primary key, so version ordering degraded to lexicographic" becomes, at both coarser levels, only the word **"sorting"**. The identifiers `pkgLabel`, `@version` and the term `lexicographic` are all gone.
- *Rules are the most fragile of all.* Reusable rules ("check the API docs before changing a builtin's semantics") almost never survive as rules; they leave a trace only inside an unrelated todo ("sync docs-site").

Two qualifications. First, quantities that are **used to state a conclusion** are dropped, while quantities that function as **status or acceptance criteria** often survive — so it is not "numbers are lost" but "evidentiary numbers are lost". Second, this ordering holds only where compression actually happens (§5): in the single-input case nothing is dropped because nothing is compressed.

6.3 An operational finding: a summary failure left a lasting gap

The deployment's week summaries covered 4 of 6 closed weeks. The two missing weeks (09-07 and 09-14) were not randomly absent: a scheduler bug in the same period (compact entries exceeding the endpoint's 32768-token context → HTTP 400 → failed → unbounded retry) meant that for two days the summarizer produced **zero** records, while the work-log path was unaffected. Normal weeks are generated at Monday 00:10–00:20; those two never appear at that cadence.

Because the scheduler only summarizes the most recently closed window, the gap persisted. Backfilling the two weeks (an explicit `mem/summarize` per window) restored week coverage to 6 weeks and moved the chain measurement from 0/6 to 3/6 recoverable.

We report this because it is the kind of evidence a synthetic benchmark cannot produce: **a failure in the memory system produced a measurable, persistent gap in its own memory**, and the shape of the gap (bounded to the failure window, day summaries intact) identifies the cause.

---

7. Methodology: two traps and a negative result

7.1 Trap 1 — a length artefact that inverted the conclusion

Comparing the whole day's summaries against the whole week's summaries gave the week level *higher* retention than the day level (42% vs 29% for quantities). A coarser level being more faithful than its input is impossible as a compression claim, and the cause was arithmetic: the week bundle is ~3.4x longer and covers six more days, so a

token is likelier to appear there by coincidence. Restricting both sides to the same project and window reversed the ordering (§5).

Generalisation. Any retention metric computed over texts of different sizes and scopes is suspect. The tell is an ordering that contradicts the mechanism.

7.2 Trap 2 — an extraction ambiguity that manufactured failures

Probes originally located their answer by (topic, day). A summary's prose conclusion frequently names several topics and several options in one sentence ("keb and vun were applied options A and C"), so a reader taking the first option letter returns another topic's answer — and the wrong answer is *plausible*, so it does not look like a bug. Every miss the layered backend showed traced to this; with per-fact opaque reference tokens the backend scored 100%.

7.2b Trap 3 — packed segments, where the reader returns a neighbouring fact's answer

A summary routinely compresses a day into one line: `- Day 1: ZT02 选方案 B (keb), ZR01 选方案 A (rix)`. A reader that takes the segment's first option answers `ZR01` with `B`. The letter is a valid option in the same segment, so the miss looks like a wrong answer rather than a reader bug — and because summaries pack facts *more* densely as the level coarsens, the artefact grows with the level being measured. It made the coarse arm score 0% at every lag up to six days while the correct options were present in the text.

The fix is positional: locate the reference token and read the option *after* it (or, when the token follows its option, the nearest one before it).

Generalisation (both traps 2 and 3). When a backend's failures cluster on one answer value, or when they appear at one level and not another with the right answer visibly present, suspect the reader before the system.

7.3 Negative result — LLM judges overstate loss

Judging whether a fact "survived" is not mechanically defined, so we ran independent LLM judges over the same material. Their outputs are confident and detailed. Of the specific claims we verified in depth, **two of two were wrong**:

judge claim	verification
Week summaries for the same week are byte-identical (same md5, 15796 chars)	☑ **holds**
The week summary is "the last day's day-summary set"	✕ **refuted** — per-day comparison finds only 0–1 groups verbatim-identical
A given day's summary "omits the chronomem group entirely; the whole causal chain and all figures are gone"	✕ **refuted** — the group is present and complete, including the root-cause chain and the terms `OOM`, `sseCond`, `CLOSE_WAIT`, `heartbeat`

Both refuted claims erred in the same direction: asserting absence that was not there. The judge's *counts* (intact / partial / absent distributions) are likewise not citable for this reason.

Judges remain useful for one thing: they **pointed at the right phenomena** — the byte-identical week output was real, and finding it led to the per-input-size analysis of §5. But the direction and the numbers had to come from mechanical re-measurement.

7.4 What we therefore rest on

Every quantitative claim in §4–§6 comes from a script that can be re-run and whose output is archived with the repository: the seeded trace generator, the four-backend runner, the ablation, the token-survival and week-step measurements, and the chain checks. The judge outputs are archived too, marked as direction-only. The harness itself has 31 regression tests plus a mutation check that deliberately breaks the implementation in ten ways (including the ones above) and fails if the suite does not notice — added after the traps of §7.1–6.2, precisely because passing tests prove little if the tests never exercise the thing they name.

---

8. Limitations

- **Three seeds.** The synthetic results rest on seeds 7/11/23. Exact recall's zero variance across buckets makes its 100% robust; the summary figures carry bucket standard deviations up to 38% and should be read as centres, not point estimates. Summary buckets hold 2–14 probes each.
- **Synthetic trace.** Mechanism-level claims (capacity cliff, destructive compaction) transfer; magnitudes calibrated to this trace do not.
- **No ground truth on the real corpus.** §6 is survival measurement and blind recall, not accuracy. §6.1 shows that what was recorded is recoverable; it cannot show that everything important was recorded.
- **Judge coverage incomplete.** Of four planned judge shards (24 days), two produced output (12 days). Since judge counts are not citable (§7.3) this affects only the qualitative colour, not the reported numbers.
- **Single deployment, single user, ~2 months.** No multi-tenant, authenticated or encrypted operation was exercised; scale is thousands of records across a handful of types.
- **Only the `day → week` hop was measured.** The service also supports finer levels (`10min` / `30min` / `1h` / `3h`); their behaviour is not characterised here, and the traces used here declare `[day, week]` only.
- **The level chain ends at `week`.** This bounds what "longer spans" can mean. The service recognises a `month` level but no `quarter` or `year`, and the production `worklog` type enables neither; a span beyond the coarsest enabled level is not summarised more coarsely, it is covered by **exact recall plus pagination** over the raw series. Our measurements are consistent with that being sufficient rather than a stopgap: exact recall answered 100% in every lag bucket up to 29 days, and the reader pages automatically (measured: 300 records over 2 pages, no truncation). What remains unmeasured is a span long enough that the *raw* series itself becomes impractical to page through — which is a volume question about the deployment, not a ceiling of the summarisation hierarchy.
- **One implementation of compaction.** The one-shot baseline is our own; a stronger compressor would shift its numbers, though not the structural argument that it cannot answer about content it has discarded.

---

9. Conclusion

We set out to execute the position paper's evaluation plan and found that the most interesting result was about the measurement.

The comparison part is unambiguous: a bounded recency window answers perfectly within its capacity and then not at all, with the cliff tracking capacity across a 6-point ablation; one-shot compaction scores zero even at the shortest lag; layered exact recall holds 100% across the full span with zero variance. The cost asymmetry the position paper

argued for is real and measurable: the no-LLM path alone answered 6/6 blind questions about two-month-old operational facts.

The recursive-summarization part required correcting our own method three times — the measurement calibre, then a claim we had overstated, then the token detectors the claim rested on — and the corrected result is more specific than the original. **The step's fidelity is governed by input diversity, not by the level.** When a project contributes a single day, the step usually declines to compress: 68% of such pairs come back at ≥0.95x length, many character-identical to the input, so the apparent 85% retention mostly means the step did not attempt the work (19% of single-input pairs do compress substantially, which is why this is a tendency and not a rule). When several days are merged, the text halves and roughly 40% of quantities and half the causal connectives disappear in one step. A "compression ratio per level" is not a property of the architecture. On real data the ordering of losses is stable: evidentiary numbers first, then mechanism, with identifiers lasting longest — and the fragility of reusable rules is the pattern most likely to matter in practice, since rules are the part of experience that is supposed to generalise.

Finally, a negative result we did not expect to be reportable: independent LLM judges produced specific, confident, wrong claims — both of the two we verified in depth asserted absence that was present. We report it because the same failure mode will affect any evaluation of memory systems that relies on model judgement of "did this survive": the judge cannot see what it has been given only in summary form, and neither, without care, can the evaluator.

References

The position paper's reference list applies, and its memory-system entries (MemGPT, Generative Agents, HiMem, RaMem, AriadneMem, A-MEM, Mem0, Letta) are the context for §2. Added here are the works §2 compares this paper's measurements against, plus the artefacts that make the numbers reproducible. Every entry below was verified against the publisher, not recalled.

1. Wang, Q., Fu, Y., Cao, Y., Wang, S., Tian, Z., Ding, L. *Recursively Summarizing Enables Long-Term Dialogue Memory in Large Language Models.* Neurocomputing, 2025; arXiv:2308.15022. — the mechanism §5 measures, and the closest prior work to it.
2. Jiang, D., Li, Y., Li, G., Li, B. *MAGMA: A Multi-Graph based Agentic Memory Architecture for AI Agents.* Proceedings of ACL 2026 (Long Papers), pp. 36848–36865. DOI: 10.18653/v1/2026.acl-long.1709. — structure-preserving memory, from the design side.
3. Huang, Z., Tian, Z., Guo, Q., Zhang, F., Zhou, Y., Jiang, D., Xie, Z., Zhou, X. *LiCoMemory: Lightweight and Cognitive Agentic Memory for Efficient Long-Term Reasoning.* Findings of ACL 2026, pp. 36842–36858. DOI: 10.18653/v1/2026.findings-acl.1835. — hierarchical graph memory over entities and relations.

Artefacts:

4. Position paper: *Chronomem (忆时塔): Time-Layered External Memory as an MCP Service for Agent Recall and Summarization.* AiraXiv 2610.0035.
5. Evaluation harness and archived results: `chronomem/bench/` and `chronomem/bench/results/` in the implementation repository (`https://atomgit.com/ystyle/chronomem`). The scripts named in this paper — `retention_eval.py`, `week_step.py`, `gen_trace.py`, `stats_intervals.py`, `validate_classifiers.py`,

`semantic_survival.py`, `rouge_survival.py`, `variance_bench.py` — and their outputs under `bench/results/` are what every quantitative claim resolves to.

6. Consolidated evaluation documents: `chronomem/docs/evaluation-retention.md`, `chronomem/docs/evaluation-real-corpus.md`, `chronomem/docs/evaluation-baseline.md`.
7. Incident records. The serving misconfiguration in §3.5: `chronomem/docs/incidents/2026-10-16-embedding-pooling-mismatch.md`. The scheduler failure in §6.3 — compact entries pushing a grouped prompt past the endpoint's 32768-token context, giving HTTP 400 and an unbounded retry — is recorded as a lesson in the deployment's own memory rather than as a file: retrieve it with `mem/query {type: "dsh/lesson", tag: "topic=llm-context-budget"}` on the running service. We cite it this way deliberately, since a memory system holding the record of its own failure is the §6.3 point.

AI-Use Disclosure

The implementation, the evaluation harness and the analysis were developed in collaboration with an AI assistant (DeepSeek). Independent LLM judges were used as a *foil*: their outputs are reported as a negative result in §7.3 and are not the source of any quantitative claim. All reported numbers are produced by scripts archived with the repository and were re-verified against that output during writing.