The Compaction Half-Life: Measuring Fact Survival Under Iterated Context Summarization

Yash Singh
IIIT Una

July 19, 2026

Abstract

Long-running LLM agents periodically compress their working context into a summary and continue from the summary alone. I measure how fast discrete facts decay under this iterated compaction. In each chain, 16 atomic facts (8-character codes with plain-language descriptors) are planted in a roughly 3,000-word task log; the model compresses the log to a word budget, receives roughly 1,200 words of fact-free distractor activity, and re-compresses, for 5 rounds. I run 3 pre-registered arms (6 chains each, 90 calls) on a small production model: a 150-word budget, a 400-word budget, and a 150-word budget plus a one-line instruction to preserve identifiers verbatim. Retention decays quickly under plain instructions: round-5 survival is 0.031 (150-word arm) and 0.083 (400-word arm), with fitted per-round survival $s = 0.484$ and $s = 0.573$, i.e. a compaction half-life of roughly 0.95 and 1.24 rounds. The verbatim-preservation instruction dominates both: all 96 planted codes survive all 5 rounds ($S(5) = 1.000$), a pre-registered difference of $+0.969$ round-5 survival over the plain 150-word arm (95% CI $[+0.906, +1.000]$). The 2.7× larger budget alone produced no claimable round-5 difference. Loss is availability, not corruption: 0 near-miss mutations of planted codes were observed in any summary, and in probe tests lost facts led to abstention (17/17), never confabulated codes. A one-line prompt change, not a bigger summary budget, was the decisive lever for fact survival under compaction.

1 Introduction

Agentic LLM systems outlive their context windows. Coding agents, support agents, and long-horizon research agents all eventually compress their working history into a summary and continue from the summary alone, often many times in a single session. Each compaction is lossy, and losses compound: a fact dropped at round 2 is unavailable at every later round. Prior work has documented how performance degrades as context grows [2] and how models lose information across multi-turn interaction [3], but the decay rate of specific, verifiable facts under *iterated* summarization has received less direct measurement.

This paper asks three pre-registered questions. **H1 (shape):** does fact survival decay geometrically (a constant per-round survival probability), or is loss front-loaded in the first compaction? **H2 (budget):** does a larger summary budget (400 words vs. 150) slow decay? **H3 (instruction):** does a one-line instruction to preserve identifiers verbatim slow decay?

The design is deliberately minimal: facts are random 8-character codes attached to operational descriptors (for example, a deploy ticket reference), planted in realistic task-log filler, and scored by verbatim substring match in each round's summary. This makes retention binary, machine-checkable, and immune to scoring ambiguity. All rules, budgets, and decision criteria were fixed in a config file before the main run.

Findings in brief: decay under plain compaction instructions is severe (fitted half-life of roughly 0.95–1.24 rounds); the shape is statistically consistent with geometric decay at this sample size; the larger budget produced no claimable benefit at round 5, while the verbatim-preservation instruction produced perfect retention through all 5 rounds. Failure mode matters too: facts vanish cleanly rather than getting corrupted, and the model abstains rather than confabulating when asked about lost facts.

2 Related Work

Liu et al. [4] showed that models use long contexts unevenly, with positional (lost-in-the-middle) effects on retrieval. Hong et al. [2] measured performance degradation as input length grows even on simple tasks. Laban et al. [3] found large average drops in multi-turn settings, driven substantially by unreliability rather than aptitude, which parallels the large run-to-run variance observed here. On the constructive side, agent-memory systems compress and manage context deliberately: Mem0 [1] builds production long-term memory via extraction and consolidation, Dynamic Cheatsheet [5] maintains an adaptive test-time memory, and agentic context engineering [6] evolves contexts to improve them over time. This paper complements that line by quantifying the default loss rate such systems must beat, and by isolating a cheap prompt-level lever (verbatim preservation) that changes the decay constant dramatically.

3 Method

Chains. A chain is 5 sequential summarize-and-continue rounds. Round 1 presents an instruction plus a roughly 3,000-word synthetic task log (the source). Each later round presents the instruction, the previous round's summary, and a fresh roughly 1,200-word fact-free distractor segment describing continued activity. The model outputs only the compressed context.

Facts. Each chain plants 16 atomic facts of the form "The ⟨descriptor⟩ is ⟨CODE⟩" at controlled positions in the source. Codes are random 8-character strings from an alphabet that excludes ambiguous characters (no 0/O or 1/I/L), so substring scoring is unambiguous. Descriptors are operational (ticket references, dashboard UIDs, tokens, and similar). All content is seeded and recorded.

Arms. Three arms, 6 chains each, sharing content seeds across arms so arm contrasts compare identical facts and filler. Arm A: "Compress the following working context into at most 150 words, preserving everything needed to continue the task." Arm B: identical with a 400-word budget. Arm C: the Arm A instruction plus one sentence: "Preserve all identifiers, codes, numbers, and proper nouns verbatim."

Model and calls. The model under test is a small production model (Claude Haiku class) called through a headless CLI with JSON output; every raw response is retained on disk. The main run is $3 \times 6 \times 5 = 90$ calls. A 5-call pilot chain calibrated difficulty before the main run (pre-registered gate: round-1 retention in $[0.15, 0.95]$; observed 0.938, so the main run proceeded unchanged). Six validation probe calls and a 10-call spot-repeat (the A-0 chain spec run twice more) complete the budget. In total 105 chain calls and 18 probed facts are analyzed; total cost was USD 3.52 with mean per-call API latency 14.4s.

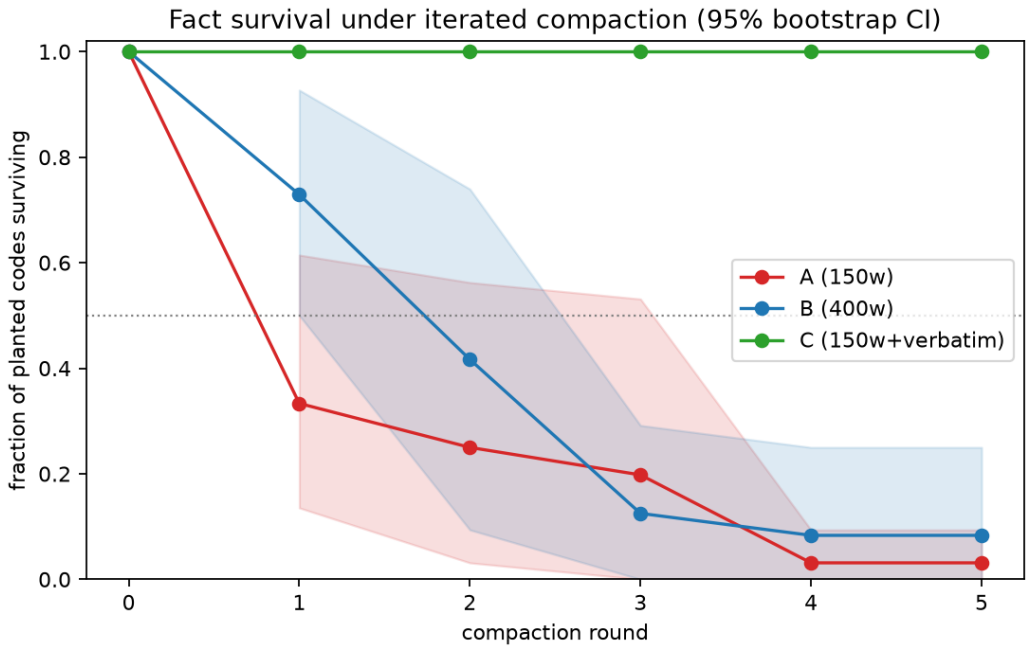

Figure 1: Fact survival by compaction round, per arm, with 95% bootstrap CIs over chains.

Scoring (pre-registered). A fact survives round r if its code appears verbatim (case-insensitive substring, whitespace normalized) in the round-r summary. Corruption is scored separately: any 6–10 character alphanumeric token in a summary at Levenshtein distance 1–2 from a planted code that is absent verbatim counts as a mutated code. Geometric fits use least squares of $\log S(r) = r \log s$ through the origin on arm-level chain-mean survival; half-life is $\ln 2/(-\ln s)$. The decay shape rule computes conditional survival $c_r = S(r)/S(r-1)$ and $\delta = c_1 - \text{mean}(c_2..c_5)$; decay is called front-loaded only if the 95% bootstrap CI of δ (resampling chains, 10,000 reps) excludes 0 with $\delta < 0$. Arm contrasts are claimed only if the 95% bootstrap CI of the difference excludes 0. All rules were fixed before the main run; every number in this paper is emitted by the analysis pipeline from the raw response files.

4 Results

4.1 Survival curves and half-life

Table 1 and Figure 1 give the headline result. Under the plain 150-word instruction (Arm A), only 32 of 96 codes survive even one compaction ($S(1) = 0.333$), and 3 of 96 survive round 5. The 400-word budget (Arm B) starts much higher ($S(1) = 0.729$) but converges toward the same floor by round 3 ($S(3) = 0.125$ vs. 0.198). Fitted per-round survival is $s = 0.484$ $[0.193, 0.589]$ for Arm A and $s = 0.573$ $[0.208, 0.733]$ for Arm B: a compaction half-life of roughly 0.95 and 1.24 rounds respectively. Arm C is at the measurement ceiling: all 96 codes survive all 5 rounds ($S(5) = 1.000$, $s = 1.000$), so its half-life is unresolvable within this horizon (reported as ∞).

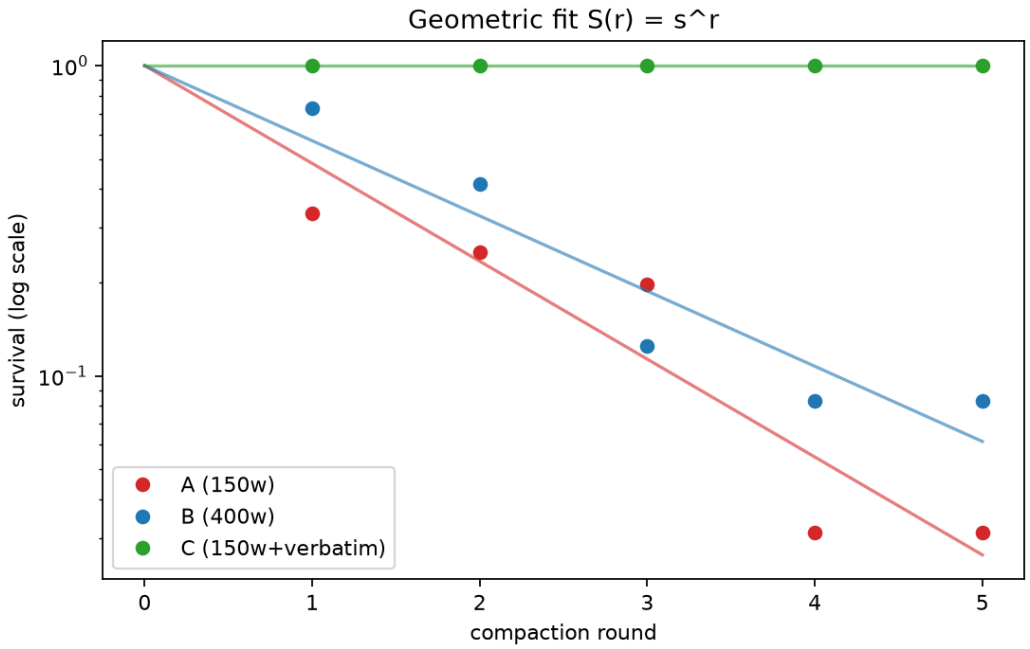

Figure 2: Geometric fit $S(r) = s^r$ on a log scale. Points are arm means; lines are the pre-registered least-squares fits through the origin.

4.2 H1: decay shape

For no arm does the pre-registered δ rule call the decay front-loaded: every 95% bootstrap CI of δ includes 0. The data are therefore consistent with geometric decay, though the wide CIs mean this is a failure to reject rather than strong evidence for constant per-round loss; Figure 2 shows visible deviations from the fitted lines at this sample size.

4.3 H2 and H3: budget vs. instruction

Table 2 reports the pre-registered contrasts. H2 fails: the 400 vs. 150-word budget difference in round-5 survival is +0.052 (95% CI $[-0.062, +0.250]$) and in fitted s is +0.089 $[-0.320, +0.460]$; neither CI excludes 0, so no budget effect is claimed at round 5 despite Arm B's clear round-1 advantage. H3 succeeds decisively: the verbatim instruction improves round-5 survival by +0.969 $[+0.906, +1.000]$ and fitted s by +0.516 $[+0.411, +0.807]$. Arm C also beats the 2.7× larger budget directly (+0.917 round-5 survival, CI $[+0.750, +1.000]$), while exceeding its own 150-word budget in only 3 of 30 rounds (mean summary length 119 words vs. 99 for Arm A and 214 for Arm B).

4.4 Failure mode: clean loss, no corruption

Across all 105 chain calls, 0 mutated codes were observed: no summary ever contained a near-miss (edit distance 1–2) variant of a planted code that was absent verbatim. Loss under compaction is availability loss (facts silently omitted), not integrity loss (facts corrupted). The validation probes reinforce this: asked questions answerable only from a round-5 summary, the model answered the substring-retained code exactly (1/1) and, for substring-lost facts, answered UNKNOWN in 17 of 17 cases with 0 code-like confabulations. The substring metric therefore tracks answerability in

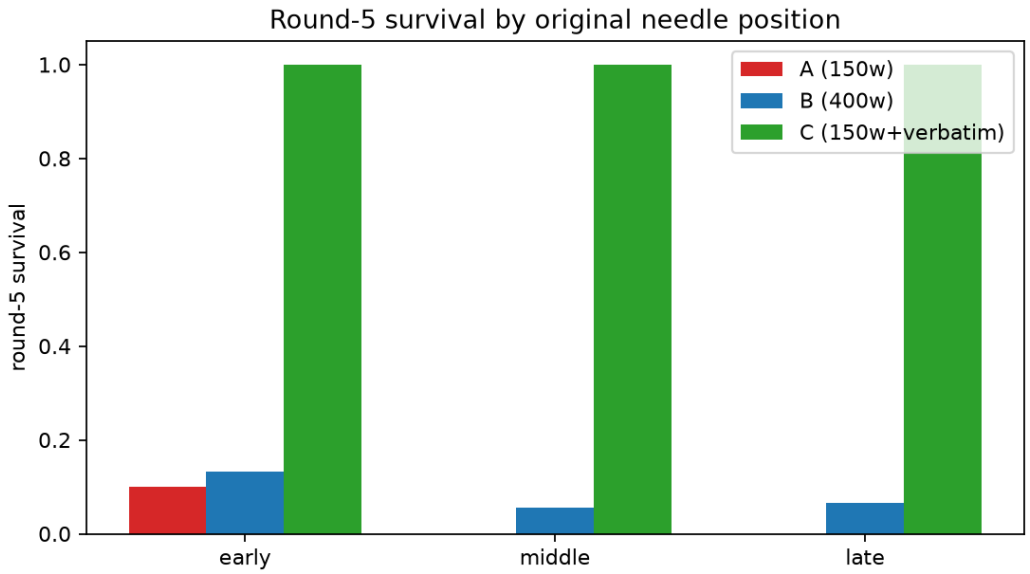

Figure 3: Round-5 survival by the fact's original position tercile in the round-0 source.

both directions, and downstream consumers of a compacted context see missing facts, not wrong facts.

4.5 Position and content effects

Survival is mildly position-dependent in the plain arms (Figure 3: early-planted facts survive best), directionally consistent with serial-position effects in long-context use [4], though this experiment is not powered to make positional claims. An exploratory, non-pre-registered observation: during the pilot the model sometimes treated credential-like values (passwords, tokens) as sensitive and redacted them during recompaction; in the main data secret-like descriptors show no round-5 survival advantage in the plain arms. The verbatim instruction eliminated the behavior in Arm C. This is flagged as exploratory only.

4.6 Variance and stability

Chain-level variance is large. In Arms A and B, round-5 survivors concentrate in single chains (one Arm B chain holds all 8 of that arm's survivors). Table 3 shows the same chain spec run three times: round-1 survival ranged from all 16 codes to none across runs. Point estimates for individual chains are therefore highly unstable, echoing the unreliability findings of Laban et al. [3]; only the arm-level contrasts with chain-bootstrap CIs (Table 2) should be trusted, and the C-arm result is the only one large enough to clear that bar.

5 Limitations

Six chains per arm gives wide CIs; H2's null is a failure to detect, not evidence of no effect. All results are from one small production model at one date through one CLI harness; other models or scales may decay differently. Facts are isolated random codes, which is the easy case for verbatim preservation; entangled or paraphrasable knowledge may behave differently, and a summary that

Table 1: Fact survival $S(r)$ by arm (mean over 6 chains, 95% bootstrap CI over chains, 10,000 reps), fitted per-round survival constant s, and compaction half-life in rounds.

Arm	$S(1)$	$S(2)$	$S(3)$	$S(4)$	$S(5)$	s	half-life
A: 150w	0.33 [0.14,0.62]	0.25 [0.03,0.56]	0.20 [0.00,0.53]	0.03 [0.00,0.09]	0.03 [0.00,0.09]	0.484	0.95
B: 400w	0.73 [0.50,0.93]	0.42 [0.09,0.75]	0.12 [0.00,0.29]	0.08 [0.00,0.25]	0.08 [0.00,0.25]	0.573	1.24
C: 150w+verbatim	1.00 [1.00,1.00]	1.00 [1.00,1.00]	1.00 [1.00,1.00]	1.00 [1.00,1.00]	1.00 [1.00,1.00]	1.000	>100

Table 2: Pre-registered arm contrasts. Difference and 95% bootstrap CI (chains resampled independently per arm, 10,000 reps); a difference is claimed only if the CI excludes 0.

Contrast	Metric	Diff	95% CI	Claimed
H2: B − A	round-5 survival	+0.052	$[-0.062, +0.250]$	no
H2: B − A	fitted s	+0.089	$[-0.320, +0.460]$	no
H3: C − A	round-5 survival	+0.969	$[+0.906, +1.000]$	yes
H3: C − A	fitted s	+0.516	$[+0.411, +0.807]$	yes
C − B	round-5 survival	+0.917	$[+0.750, +1.000]$	yes

keeps codes but drops their surrounding task state would still score perfectly here. Arm C's perfect retention is a ceiling: with 16 facts and 5 rounds this design cannot distinguish truly lossless compaction from slow decay, and sustaining verbatim retention at larger fact counts within a 150-word budget is impossible in principle. The run-to-run variance measurement uses only 3 repeats of one spec. Finally, the geometric fit is a summary statistic, not a mechanistic claim.

6 Conclusion

Under realistic iterated compaction, a small production model loses planted facts with a half-life of roughly one to one-and-a-quarter compaction rounds, and after five rounds retains almost nothing, regardless of whether the summary budget is 150 or 400 words. A single added sentence demanding verbatim preservation of identifiers moved retention from 3/96 to 96/96 at round 5 at essentially no cost in summary length. For builders of long-running agents and memory systems [1, 5, 6], the practical takeaway is that what the compaction prompt asks for matters far more than how much room the summary gets, and that losses are silent omissions, so anything not explicitly protected should be assumed gone within a few compactions.

Reproducibility. All chain specs, prompts, raw model responses, scoring code, and the analysis pipeline that emits every number and table in this paper are retained alongside the manuscript; the experiment cost USD 3.52 in API calls.

References

[1] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory, 2025.

[2] Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts LLM performance. Chroma Technical Report, 2025. URL `https://research.trychroma.com/context-rot`.

Table 3: Run-to-run stability: the identical chain spec (A-0) run three times with fresh model calls. Surviving codes out of 16 by round.

Run	r=1	r=2	r=3	r=4	r=5
A-0	16	16	16	0	0
repA0-rep1	0	0	0	0	0
repA0-rep2	4	2	2	2	2

[3] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation, 2025.

[4] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[5] Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory, 2025.

[6] Qizheng Zhang et al. Agentic context engineering: Evolving contexts for self-improving language models, 2025.