Union-Find Compaction: Provenance-Preserving Context Compression for LLM Agents

June Kim
Independent Researcher
june@june.kim
ORCID 0009-0005-3153-9396

July 5, 2026

Draft. Conversational agents usually handle a full context window by replacing old messages with one summary. The summary is compact, but it severs each claim from its source and cannot restore details it omitted. We replace that one irreversible summary with a bounded set of source-backed summaries. Related messages form clusters in a union-find forest; structural merges happen synchronously, while summary updates are coalesced and run as the main model works. This makes compaction incremental, auditable, and reversible. In a feature-flagged Gemini CLI integration over twelve GitHub-issue conversations, the method cost 0.79x as much as flat summarization without putting summary generation on the interactive path. Recall was 30.2% versus 21.9% for the baseline, a promising but non-significant difference ($p = 0.136$). Across seven controlled trials, it tied or exceeded the baseline each time. These studies do not yet prove non-inferiority or improvement; they motivate the higher-powered, formally specified replication preregistered below.

1 Introduction

Every conversational agent has the same bounded-context problem: conversations grow, but the window does not. A common response is flat summarization. The agent runs a cheap model over everything outside a recent "hot" window, compresses that material to a fixed budget, and replaces the source messages with the result. Gemini CLI compresses the oldest 70% of a session into one snapshot.

Flat summarization is useful, but the replacement loses three capabilities. A claim in the summary no longer points to the message that supports it. An omitted detail cannot be restored from the summary. And the agent cannot retrieve one relevant topic independently because the entire cold history has become a single block. In the Gemini CLI implementation studied here, batch compression also put a twenty-to-thirty-second model call on the session's critical path.

These losses come from the representation, not the quality of the summarizer. A better model may write a better paragraph, but a paragraph without source links still cannot identify its evidence or restore text that was discarded.

Our alternative keeps the sources and adds an index over them. The cold context becomes a union-find forest: each graduated message is a node, and topically similar nodes join the same cluster. The cluster root stores a summary, while membership metadata maps that summary back to its source messages. `find` locates the cluster for any message; enumerating that cluster's members recovers its evidence. With path compression and union by rank, the parent-pointer operations take amortized $O(\alpha(n))$ time, where α is the inverse Ackermann function (Tarjan 1975). This bound does not include cluster search, member enumeration, centroid updates, or model summarization.

From that substitution, three properties follow from the representation rather than the evaluation:

- Provenance. `find(m)` locates the cluster root, and the cluster's membership index identifies the source messages behind its summary.
- Recoverability. `expand(root)` reinflates a cluster to its sources. Raw messages stay addressable; flat summarization discards them.

- Incremental operation. Messages graduate one at a time; local indexing does not require reprocessing the entire cold history.

This is a replacement for flat context compaction, not a general agent-memory system. It changes how an agent compresses material leaving its active window; it does not claim autonomous long-term learning, cross-session knowledge management, or reliable retrieval from an unbounded history. The empirical question is whether the replacement preserves recall at an acceptable cost. The current results favor the forest, but the studies are too small to establish formal non-inferiority or a recall improvement.

Contributions. This work replaces one irreversible summary with a bounded set of source-backed summaries. It uses union-find as their provenance index and separates cheap structural merges from expensive model calls through deferred, coalesced summarization. A controlled Python prototype tests the clustering idea; a feature-flagged Gemini CLI integration tests the optimized design. Both provide encouraging recall results, while the field integration also reduces measured summarizer cost. A preregistered replication is designed to test formal non-inferiority and possible improvement at higher power.

2 What Flat Compaction Destroys

Treat the agent's context manager as a cache with a fixed capacity and an eviction policy. Flat summarization's policy is: when full, replace the cold region with a single summary of it. Read as a cache, that policy fails three separate contracts.

Traceability. A cache that answers from a derived value should be able to name the sources the value came from. Flat summarization cannot: the paragraph is a lossy function of the whole cold region with no inverse and no index. When the agent later asserts "the scrape interval is 30s," nothing connects that to the message that set it.

Recoverability. Eviction from a flat summary already happened, silently, at compression time; there is no way to reinflate a detail the summarizer dropped. Eviction from a structure that keeps its sources is a deferred policy choice, not an irreversible event.

Selective retrieval. A single summary is retrieved whole or not at all. There is no way to pull the one cluster relevant to the current turn while leaving the rest compressed, because there are no clusters, only the block.

The union-find forest restores all three because it never overwrites the sources; it only adds structure over them (Table 1).

	Flat summarization	Union-find compaction
Provenance	none (paragraph has no source index)	cluster membership → source messages
Recoverability	sources discarded	`expand` → reinflate cluster
Selective retrieval	whole summary or nothing	nearest cluster injected
Compaction unit	whole history, single pass	one message, incremental
Update path	batch model call on critical path	sub-millisecond local append; summaries deferred

Table 1. What each representation preserves. The evaluation asks how the structural benefits affect recall and operating cost.

3 Method

3.1 The forest

Context splits into two zones. The hot zone is the last k messages (default k = 10), served raw. When the window overflows, the oldest hot message graduates to the cold zone, a union-find forest where each cluster is one summary over its source messages (Figure 1).

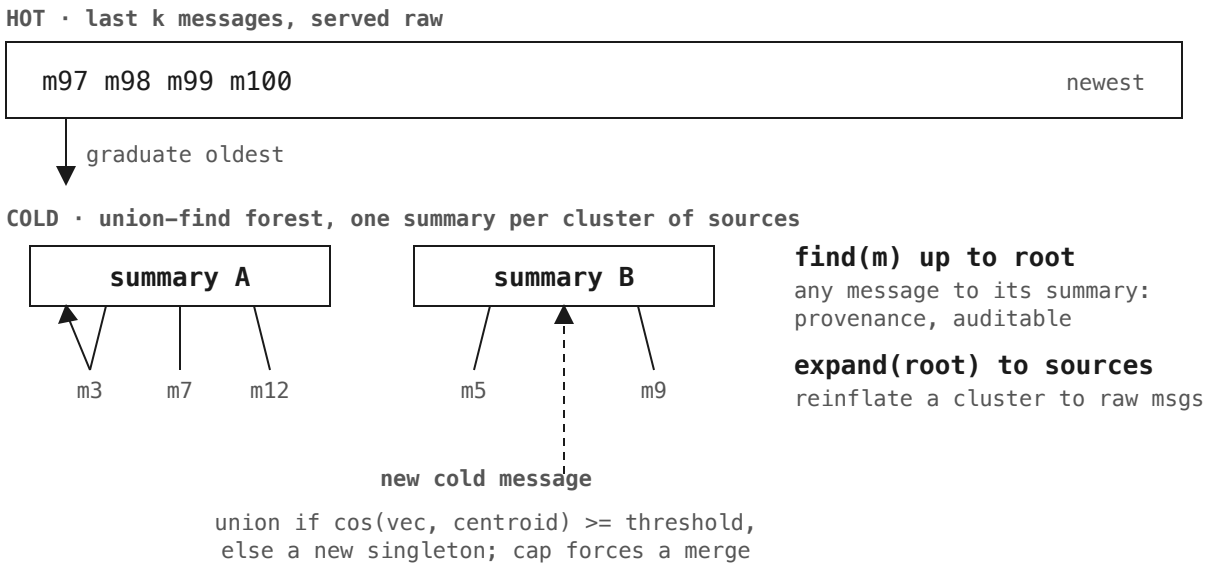

Figure 1. Messages graduate oldest-first from the hot window into the cold forest. Each cluster holds one summary and retains its source membership. *find* locates a message's cluster, *expand* returns the cluster's sources, and *union* joins a new message when it is close enough to a centroid.

The field implementation's write path, per graduated message:

1. Keep its timestamp; compute a TF-IDF vector (local, no model call).
2. Cosine-compare against cluster centroids. Above the merge threshold (default 0.15), `union` into the nearest cluster and mark its summary dirty; below, start a singleton.
3. If clusters exceed the cap (default 10), force the closest pair to merge. Centroids update as weighted averages, so the geometry stays current without re-vectorizing history.

The read path injects the nearest cluster's summary beside the hot window. Unlike retrieval systems that generate or rerank passages with a model at query time, this prototype chooses among summaries prepared in advance. Injected context therefore stays bounded and retrieval needs no model call. The forest provides cluster lookup and incremental updates; retained source storage and a membership index provide expansion and auditability.

3.2 Deferred, coalesced summarization

Union-find makes parent-pointer updates cheap; it does not make summarization cheap. The naive prototype re-summarizes a cluster from its full membership after every merge. If one cluster grows one message at a time, successive calls read 2, 3, … n messages: quadratic source-text volume overall. In an early build, this produced roughly 80 summarizer calls per conversation against flat summarization's 2 and a 5.2x cost premium.

The field implementation decouples the two operations. `union` updates parent pointers, membership, and centroids synchronously, then marks the root dirty. For each dirty root, the system retains its last clean summary plus the raw messages added since that summary. A background resolver coalesces all intervening merges into one call:

```
summarize([last_clean_summary, ...new_messages])
```

Thus each newly graduated message enters one summarization batch rather than every later re-summarization of its cluster. The bounded previous summary carries older information forward. Multiple dirty clusters can resolve concurrently, and generation guards prevent an in-flight result from overwriting a root that changed during the call.

Resolution runs while the main agent model is already working. An overlap window keeps newly graduated messages available verbatim until their cluster summary is ready; `render` uses cached clean summaries and does not await a summarizer. This moves model latency off the interactive path rather than pretending

the model call is sub-millisecond. In the field study, coalescing and smaller prompts turned the early cost regression into a cost improvement (see §4.2).

This optimization changes summarizer input growth, not the asymptotic cost of every operation. `find` and the parent-pointer portion of `union` are amortized $O(\alpha(n))$; expanding a k-message cluster is O(k), centroid comparison over c clusters of dimension d is O(cd), and an exhaustive closest-pair fallback is $O(c^2d)$. With the tested cap of ten clusters, model inference remains the dominant cost.

4 Evaluation

The structure leaves one empirical question: at a matched token budget, how does routing compaction through the forest affect recall? The two studies below provide an initial estimate. They were not large enough to establish formal non-inferiority; that is the purpose of the planned replication.

4.1 Controlled study

The controlled study uses the original, eager Python prototype: a synthetic 200-message DevOps conversation seeded with 40 verifiable facts. It tests whether topical clustering preserves recall, not whether deferred summarization reduces cost. Both methods use the same cheap summarizer (Haiku), the same token budget, and the same retrieval machinery. An LLM judge scores binary recall: "PostgreSQL 16.2" counts, "PostgreSQL" does not. McNemar's test is applied to the discordant pairs. Seven trials vary the summarizer, compression ratio, retrieval, tuning, and timestamping.

#	Config	Flat	UF	p
1	Haiku, 50	90%	90%	1.000
2	Haiku, 200	65%	82%	0.039
3	Sonnet, 200	70%	78%	0.453
4	Haiku, 200, retrieval	68%	82%	0.180
5	Haiku, 200, tuned	62%	80%	0.065
6	Haiku, 200, timestamps	72%	90%	0.092
7	Haiku+Sonnet, 200	75%	90%	0.070

At low compression (50 messages), the methods tie. At 200 messages, union-find is higher in every trial by 8 to 18 points. Trial 2 reaches $p = 0.039$; the remaining trials are individually inconclusive at $n = 40$ facts. Because the trials change several conditions rather than repeat one fixed experiment, their consistent direction is suggestive but does not constitute a pooled significance test. Trial 7 most closely mirrors production: a cheap model summarizes and an expensive model answers. The expensive answerer cannot recover a fact already omitted during compaction, so recall must be preserved at the summarization stage.

4.2 Field study

We implemented the method in a feature-flagged fork of Gemini CLI and evaluated it on 12 GitHub-issue conversations of about 120 messages each. Eight factual questions per conversation (96 total) were generated from the uncompressed content and scored by a blinded LLM judge. Flat compression ran on the same data. We preregistered three hypotheses before any run.

Hypothesis	Result	Detail
Latency	PASS	append p95 = 0.33ms, render p50 = 0.006ms
Cost	PASS	0.79x flat, 21% cheaper
Recall	Inconclusive	+8.3pp (30.2% vs 21.9%), $p = 0.136$

Latency and cost pass their preregistered thresholds. Deferred summarization makes 35 summarizer calls across 12 conversations, compared with 24 for flat compression and 960 projected for the eager forest under the same workload. Its 11 additional calls use smaller prompts, bringing total summarizer API cost to 0.79x the baseline. That figure excludes local TF-IDF vectorization, cosine similarity, centroid updates, and serialization. Append and render bookkeeping are sub-millisecond; background resolution is not, but overlaps the main model call instead of blocking the next turn.

The recall result is encouraging but inconclusive. Union-find leads by 8.3 points across 96 questions (30.2% versus 21.9%), winning 8 conversations, tying 2, and losing 2. The difference is not statistically significant

($p = 0.136$). A non-significant two-sided test cannot establish either equality or non-inferiority, so this study supports neither a "no regression" claim nor a positive improvement claim on its own. The point estimate supplies an effect size for the formally specified replication in §6.

The integration is public and inspectable to any depth: the implementation submitted as PR #24736 (not merged), the issue, the design discussion, and the preregistration, raw data, and latency CSVs in the spec repository.

The platform's sunset dates the artifact, not the approach: Google announced on May 19, 2026 that gemini-cli would transition to Antigravity CLI, stopping service for free and individual users on June 18, 2026, and this study predates both. The method is not specific to gemini-cli; any agent that compacts by flat summarization can adopt the same forest.

4.3 Why the footnotes survive

Where union-find leads, it leads on footnote facts. Flat summarization preserves headline facts (the database version, the auth scheme) and drops the scrape interval, the cron schedule, the webhook path, the filterable-attribute count: the details that separate having read the conversation from having read a briefing. The mechanism is competition. Flat summarization compresses the whole history in one pass, so every fact competes for space in a single budget and footnotes lose to headlines. Union-find compresses per cluster, 5 to 20 messages each, so the cron schedule is summarized alongside its neighbors, not against the database version. Facts compete only within their cluster, and most footnotes are the headline fact of some small cluster.

5 Related Work

Flat summarization is the deployed baseline in production agents (Gemini CLI, and comparable context managers) and the method we measure against. Its known failure, dropping detail under a single-pass budget, motivates a structured alternative.

Structured eviction and provenance memory. A concurrent line replaces single-summary compaction with structured eviction and provenance-tagged agent memory. These share the diagnosis: flat summarization drops fine-grained facts, and provenance should be preserved. They differ in the primitive: none uses a disjoint-set forest with equivalence-class merging as the compaction structure, which is the specific contribution here.

Submodular and diversity-aware selection. Context selection by submodular or determinantal objectives targets a related goal (keep a diverse, non-redundant subset) but selects among items rather than merging them into canonical, re-expandable clusters, and does not provide provenance back to sources.

Retrieval-augmented generation retrieves raw passages per query at read time; union-find pre-merges at write time, giving bounded injected context and no per-turn retrieval call, at the cost of committing to a clustering online.

Long-horizon memory systems. Systems evaluated on multi-session benchmarks such as LoCoMo and LongMemEval solve a broader problem: selecting and maintaining information across many sessions. Union-find compaction operates inside a bounded context manager and is evaluated against the flat compactor it replaces. Memory benchmarks may provide useful stress cases, but they are not the primary product category or evaluation target.

Union-find itself is Tarjan (1975); the contribution is not the algorithm or its complexity bound but its use as a provenance spine for context compaction, where `find` supplies message-level lineage and `union` supplies single-message incremental merge.

> Citations in this section are drawn from a prior-art sweep and name concurrent work by topic; the specific arXiv identifiers must be verified against the sources before submission. An earlier form of this work appeared on the author's blog and is the method's only prior public description; a preprint must cite it as such and clear the venue's prior-publication bar.

6 Limitations and Planned Confirmation

6.1 What the current evidence shows

Union-find is tied or higher in all seven controlled trials and leads by 8.3 points in the field study. One controlled trial reaches $p < 0.05$; the field study does not ($p = 0.136$). These results justify further testing, but they establish neither improvement nor non-inferiority. The latter requires a prespecified margin and confidence interval, not a non-significant two-sided test.

6.2 The preregistered confirmation

The preregistered replication holds the design fixed and scales to 200 or more paired questions across a larger conversation corpus, powered to detect an 8-point difference at the observed base rate. It also adds contradictory and stale facts, where a later message overrides an earlier one and compaction must keep the correction traceable rather than blend the two. The hypothesis, design, and analysis are fixed in advance; only the token budget to run it is pending. If the recall result does not replicate, the representation will still preserve source membership, but its empirical tradeoff against flat summarization will remain unresolved.

6.3 Further limits

The evaluation data is a proxy, not the deployment distribution. The controlled study is a synthetic DevOps conversation and the field study is scraped GitHub issue threads; neither is an agent-native session, which interleaves tool output, code, failed attempts, corrections, and agent turns at a density and structure a human issue thread lacks. Issue threads share the property under test (facts scattered across messages, footnotes competing with headlines), so they are a fair initial testbed, but whether the recall and cost results transfer to real agent context is untested. The next evaluation should therefore replay agent-native traces through both compactors at the same rendered-context budget, with source-tagged questions scored mechanically where possible. Contradictions, superseded values, tool output, and topic recurrence should be explicit strata. Established memory benchmarks can supply secondary stress cases, but they do not replace this direct test of the compaction boundary. Judge leniency remains a weak point of the current field study.

Three more. The cost win is measured at one scale (about 120 messages, ten clusters); union-find makes more and smaller summarizer calls than flat's few large ones. At much longer conversations, the crossover point where fixed per-call overhead would erase the saving is uncharacterized. The evaluation compares against flat summarization only, not against the concurrent structured-eviction systems of §5, so the evidence shows union-find beats the deployed baseline, not that it beats the nearest structured alternative. And storage grows monotonically: the forest only adds nodes and the source store accumulates, so a long-lived deployment eventually needs a cluster-eviction or archival policy, deferred here. The merge threshold, cluster cap, and hot-window size are fixed defaults whose effect on recall is not characterized.

7 Scope and Possible Extensions

The measured system ends at the context-compaction boundary. Its job is to decide what replaces messages that no longer fit in the active window and to keep those replacements linked to their sources. Serialization is useful for restoring the compactor's own state after a restart, but this paper does not evaluate cross-session recall, shared team memory, schema formation, or concurrent knowledge stores.

Source-backed clusters could become a substrate for those systems, but each would add requirements absent here: retention and access-control policy, conflict resolution, cluster splitting, and evidence-aware retrieval over a much larger store. They should be evaluated as separate designs. The broader shared-understanding proposal is developed in Union Found; it is not part of the contribution claimed here.

8 Conclusion

Flat compaction turns the cold history into one irreversible paragraph. Union-find compaction replaces it with a bounded set of source-backed summaries that can be audited, expanded, and updated incrementally. In the Gemini CLI field study, the implementation used 0.79x the baseline's summarizer cost and overlapped summary generation with the main model call. Recall favored the forest in both studies, but the samples do not establish improvement or non-inferiority. The preregistered replication is intended to decide that empirical question. The claim is deliberately narrow: this is a practical alternative to flat compaction, not a complete memory architecture.

9 Availability

Code, data, preregistrations, and result logs are archived on Zenodo: the controlled study at 10.5281/zenodo.21215158 (union-find-compaction) and the field study, with its three preregistrations and raw results, at 10.5281/zenodo.21215160 (union-find-compaction-for-gemini-cli). The gemini-cli implementation was submitted as PR #24736. Released under CC BY-SA 4.0.

LLM collaboration disclosure

LLMs enter this work in three roles. Subject of study: the compaction method drives shipped LLMs as its summarizer (Claude Haiku 4.5) and, in the production-mirroring trials, its answerer (Claude Sonnet 4.6), and recall is scored by a blinded LLM judge; the model versions are recorded in the archived result logs (§9). Instrument: independent model families adversarially reviewed the draft and its claims, and the recall judge scores each answer against the uncompressed source, with the mechanical layer (McNemar's test, the harness) holding the verdict. Writing aid: the prose was drafted and revised with Anthropic's Claude (Opus 4.8) from the author's blog posts, experiment logs, and direction; the method, the experiments, the numbers, and the argument are the author's. No LLM decided what to publish.

Funding

This work was conducted independently, with no external, institutional, or commercial funding. All compute and model-API costs were borne by the author.