VERIMEM: SERVING-TIME VERIFICATION FOR MUTABLE AGENT MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Persistent memory lets language-model agents reuse information across interactions, but source updates can leave cached summaries and plans obsolete. Rejecting an entire derivative prevents stale reuse at the cost of discarding unaffected information. We introduce VeriMem, which records source invalidation and checks versioned dependencies before memory enters model input, either rejecting unsupported derivatives or reconstructing their admissible components. Under explicit lineage and serving-state assumptions, these rules provide conditional exclusion and retention properties. Lifecycle tests eliminate observed stale-derivative exposure, while reconstruction satisfies 32/32 answer contracts versus 8/32 for whole-artifact rejection on 32 inputs from 16 related worlds, matching equally informed component baselines. With Qwen3.5-9B Q4, verified contexts raise accuracy by 16.7 percentage points on 60 previously seen LongMemEval-V2 questions with injected stale content, while reconstruction yields 56 current-answer matches versus 40 for rejection across 72 related MQuAKE questions. These results support serving-time verification as a complement to eager maintenance, combining invalid-content exclusion with selective retention to improve answers in the evaluated settings.

1 INTRODUCTION

Persistent memory allows language-model agents to reuse information across interactions (Park et al., 2023; Packer et al., 2023; Zhang et al., 2024). Besides source records, agents retain derived summaries, recommendation caches and plans. These artifacts have their own identifiers and lifetimes: correcting a source need not update an independently cached derivative. A current source in storage therefore does not make an old prompt fragment current. The question is not only whether retrieved information is relevant, but whether its support remains valid when it becomes model input.

Figure 1 illustrates a preference update from Diet@1 (seafood) to Diet@2 (vegan), while Budget@1 remains 50. An independently cached plan still depends on Diet@1. Unchecked serving passes the old preference; rejecting the whole plan also loses the valid budget. Component reconstruction offers a third outcome: preserve the budget without passing the retired preference or inventing its replacement.

Dependency tracking is established in truth maintenance, materialized views and build systems (Doyle, 1979; Gupta & Mumick, 1995; Mokhov et al., 2018). Eager invalidation follows dependencies to mark affected derivatives when a source changes. VeriMem instead records source invalidation and checks each retrieved candidate before use. Both perform update-time bookkeeping; neither requires immediate deletion or rewriting. Checking at use can avoid advance maintenance of caches that are never requested, but trades that work for read-time verification. Independently of timing, component-level policies can preserve information that whole-artifact rejection loses. We therefore ask: *after a source changes, can we exclude invalid derived content while retaining unaffected information, and how does that retention affect answers?*

VeriMem is a verification layer between retrieval and context construction. It binds derivatives to source versions and checks their current lifecycle state. Whole-artifact Gate withholds a candidate when any recorded support fails. Optional reconstruction selects explicitly supported components, assembles the retained claims and verifies the result. This selection-and-assembly procedure reuses supplied evidence. Under stated lineage, snapshot and rendering assumptions, we establish conditional exclusion and retention properties. Building on established dependency tracking, VeriMem specifies

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

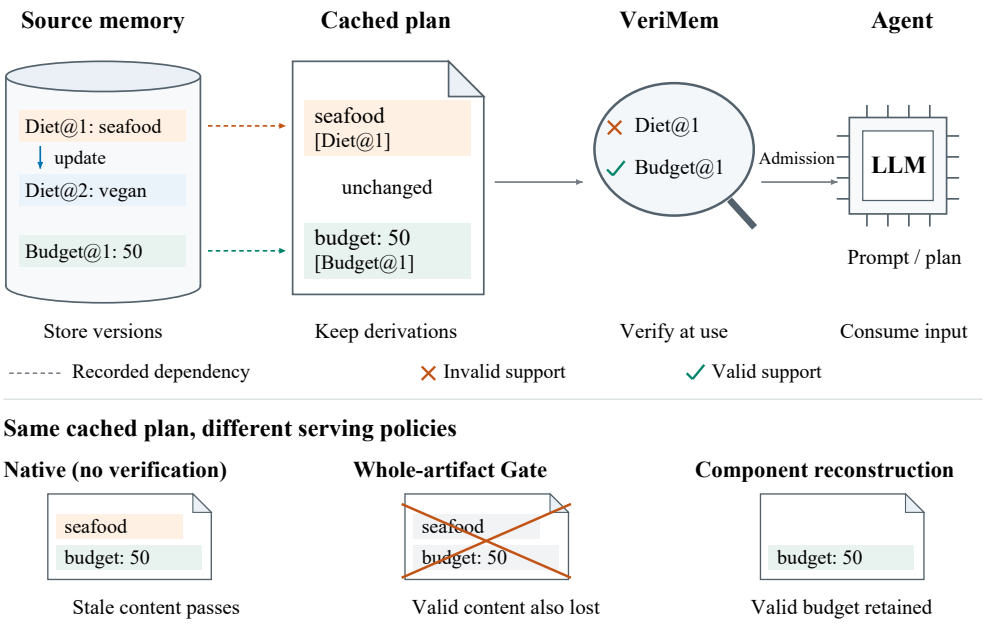

Figure 1: Source change and three serving outcomes. Name@k denotes source version k; dashed arrows indicate recorded dependencies. The cache still binds to retired Diet@1 and unchanged Budget@1. Native passes both claims, Gate withholds the whole artifact, and reconstruction retains and re-verifies the budget.

an explicit admission contract for derived agent memory and evaluates its effects on exclusion, retention and answers.

Our experiments separate three questions. Backend lifecycle tests and public-text interventions test exclusion of retained stale derivatives. Matched-information tasks test component retention: reconstruction satisfies 32/32 answer contracts versus 8/32 for whole-artifact rejection, tying strong component baselines. Finally, the LongMemEval-V2 stale-content comparison improves Qwen accuracy by 16.7 percentage points on 60 development questions. On selected MQuAKE cases, Qwen produces 56 current-answer matches with reconstruction versus 40 with Gate and 12 with unchecked serving across 72 related questions. These comparisons evaluate whether retained evidence helps this reader, separately from whether its support is valid.

The paper makes two contributions:

1. A serving-time admission contract for versioned derived memory, with whole-artifact verification, optional component reconstruction, and conditional exclusion and retention properties.
2. An evaluation separating invalid-content exclusion, valid retention and answer utility. Equally informed baselines isolate granularity from maintenance timing; Qwen comparisons quantify answer gains from exclusion and finer retention in these settings.

2 Related Work

Persistent memory and knowledge updates. Retrieval-augmented generation supplies external knowledge during prediction (Lewis et al., 2020). Generative Agents and MemGPT extend reuse across interactions through stored experiences, reflections and hierarchical memory management (Park et al., 2023; Packer et al., 2023); A-Mem, Mem0 and Zep organize linked, consolidated or temporal memories (Xu et al., 2025; Chhikara et al., 2025; Rasmussen et al., 2025). Updating these stores is itself an active problem. ERASE deletes or rewrites existing entries as new documents arrive (Li et al., 2025), while TEPA revokes keyed precedents contradicted by fresh evidence, preserving their history (Zhou et al., 2026). These approaches motivate examining not only source updates but

also the independently cached derivatives later presented to a reader. LoCoMo and LongMemEval evaluate long-term memory (Maharana et al., 2024; Wu et al., 2025; 2026); our adaptations add explicit dependency and lifecycle interventions.

Dependency maintenance and memory repair. Truth maintenance tracks reasons for beliefs as assumptions change (Doyle, 1979). Materialized-view maintenance, data provenance and build systems likewise connect derived state to its sources and organize updates (Gupta & Mumick, 1995; Buneman et al., 2001; Mokhov et al., 2018). They establish precedents for invalidation and selective reuse. Complete mediation and execution monitoring provide foundations for enforcing checks at use (Saltzer & Schroeder, 1975; Schneider, 2000). For generated memory, these ideas motivate explicit bindings between claims and their supporting source versions.

Recent memory methods overlap directly with this setting. MemoRepair withdraws affected descendants before repair and selects validated, predecessor-closed successors using a min-cut formulation (Zhao et al., 2026). PACMI propagates validity through typed dependencies and evaluates memory, context and answer outcomes (Wang et al., 2026b). Shen et al. (2026) study revoked records returned by memory systems and introduce a read-time guard. Building on these precedents, VeriMem focuses on which components of a retrieved derivative may enter the current model context. It assembles a reconstructed artifact for immediate admission and checks the support of individual components, extending the check from a returned record's own status to its dependencies. This focus on context construction complements store-wide repair and republication. The appendix compares these overlapping contracts conceptually. Our experiments isolate granularity using component-policy baselines.

Evidence selection and downstream answering. Evidence validity and relevance contribute separately to useful context. Irrelevant information can distract language models (Shi et al., 2023), and the position of relevant evidence can affect its use (Liu et al., 2024). Self-RAG learns when to retrieve and evaluates passage relevance and generation support (Asai et al., 2024). Post-retrieval assembly separates semantic evidence extraction from answer-policy execution (Reddy & Challaram, 2026). These studies motivate evaluating both available evidence and its successful use.

VeriMem addresses a different predicate: whether a derivative's recorded support remains admissible after a source change. This support check complements relevance selection, answer verification and action authorization. We evaluate invalid-content exclusion, valid-information retention and downstream answers separately, and use equally informed baselines to distinguish granularity effects from gains specific to VeriMem.

3 Problem definition and verification objective

Memory objects and support. We study external memory whose sources may change after derivation. A source is an addressable fact or event; a derivative is an independently cached summary, chunk, answer or plan. Each recorded version has an immutable identity and a mutable lifecycle status. A new source version does not automatically become the support of an old derivative. In Figure 1, correcting Diet@1 leaves the cached plan partly unsupported while Budget@1 remains valid.

Let $G = (V, E)$ be the recorded lineage graph, where $(u, v) \in E$ means that v directly depends on u. Its support closure $S(v) = \{v\} \cup Anc(v)$ includes all transitive ancestors, including intermediate derivatives. Corrections, deletion, revocation, expiry or purpose restrictions can make a supporting version ineligible for use. For an admission request $q = (p, t)$, p is the intended purpose and t the evaluation time, not a historical date in the user's question. On a fixed serving snapshot, let I_u and P_u be the effective validity interval and permitted-purpose set of record u. Its local validity is

$$\mathrm{Valid}(u, q) = [\mathrm{status}(u) = \text{active}] \wedge [t \in I_u] \wedge [p \in P_u]. \tag{1}$$

This predicate defines lifecycle eligibility.

Serving task and objectives. Given retrieved candidates, recorded dependencies, current lifecycle state and request q, the task is to select admissible content or withhold it. A whole-artifact policy admits or rejects each candidate as a unit. With explicit component-level support, a policy may

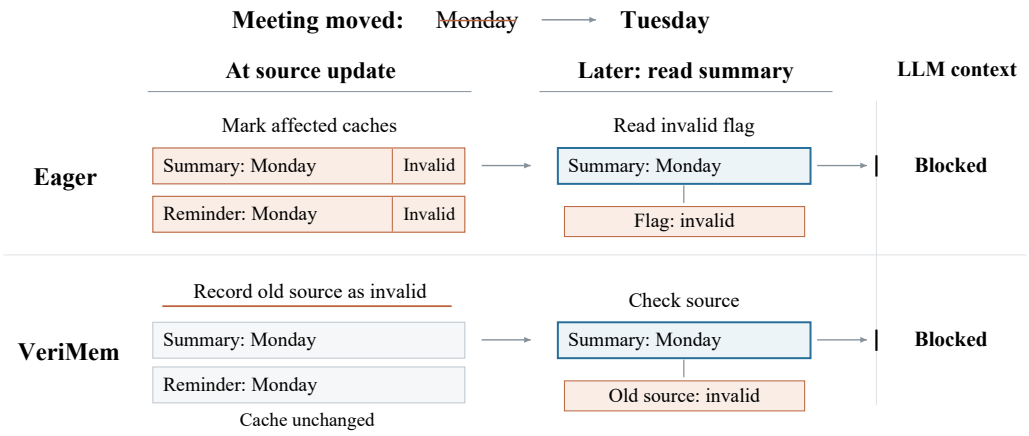

Figure 2: Update-time marking versus use-time checking. After the meeting changes, Eager marks both dependent caches; VeriMem records source invalidation and checks the requested summary before use. Both block the old summary. Blue outlines mark the requested cache; orange marks invalidation. Cached text and current support validity are tracked separately.

instead select and faithfully render a supported subset of the recorded claims $\mathcal{C}(v)$, subject to all applicable restrictions. The task concerns content supplied to the reader after retrieval.

We distinguish three objectives. *Exclusion* prevents content with invalid support from entering the reader's input. *Retention* reduces the loss of unaffected information without relaxing support requirements. *Answer utility* measures the reader's response to a task question x, distinct from admission request q. These objectives correspond to RQ1–RQ3 in Section 5. In the example, component retention preserves budget evidence that whole-plan rejection loses, while still excluding the old preference. Separate metrics measure this retained evidence and the reader's resulting answers.

Verification setting. We analyze verification with complete, accurate dependency records, a serving view reflecting accepted invalidations, and a consumer receiving only admitted content. Checks share a fixed snapshot and evaluation time; validity through consumption requires preserving that state and its time constraints until use. The studies instantiate these dependency records to examine lifecycle drift. Representation and integration requirements are in the appendix.

4 Method: Verifying and Reconstructing Derived Memory

VeriMem separates cached content from the validity of its evidence. Between retrieval and context construction, Gate verifies whole derivatives, while reconstruction selects admissible components. Both use the same support criterion, isolating the effect of selection granularity.

4.1 Versioned Support and Invalidation

Storage identity is distinct from evidential support: updating a record or assigning a new ID does not change an existing derivative's evidence. VeriMem binds writer-declared dependencies to immutable versions and tracks lifecycle status separately. A successor may support a new derivation, but does not retroactively justify an old one.

An invalidation barrier records that an affected version is blocked, even if its text remains retrievable. It must be visible before dependent content is admitted, without requiring physical deletion. In Figure 1, Diet@2 cannot justify the cached plan's old dietary claim, whereas Budget@1 remains valid. Version mapping and update coordination are detailed in the appendix.

Figure 2 contrasts timing: eager propagates invalidation at source updates, whereas VeriMem records the source barrier and checks candidates when requested. This defers work on unused derivatives but adds verification to reads. Both strategies can use either selection granularity and can be combined.

4.2 WHOLE-ARTIFACT VERIFICATION

A derivative remains supported only if every version in its dependency chain is eligible. Checking its own status alone therefore misses invalid ancestors. Given a validated graph and request, Gate checks the entire support closure $S(v)$ on the current serving snapshot and admits v exactly when

$$\begin{aligned} \mathrm{Allow}(v, q) \iff & \mathrm{ResolvedLineage}(v) \\ & \wedge \bigwedge_{u \in S(v)} \mathrm{Valid}(u, q) \\ & \wedge \mathrm{BarrierFree}(S(v), q). \end{aligned} \tag{2}$$

$\mathrm{ResolvedLineage}$ requires resolvable recorded references and acyclicity; $\mathrm{BarrierFree}$ excludes active invalidation barriers rooted in $S(v)$. The local predicate Valid is defined in Section 3. The rule deterministically tests eligibility throughout the dependency chain, complementing relevance-based retrieval. Unresolved or invalid support withholds the candidate; input validation is specified in the appendix.

The rule is conservative at artifact granularity: one invalid dependency withholds all content, including unaffected claims. Thus it excludes the old dietary preference but also loses the valid budget. This motivates the finer selection below. Each reuse receives a fresh check against the serving state.

4.3 COMPONENT-LEVEL RECONSTRUCTION

Given explicit claim-level support, reconstruction applies Gate to each claim in $\mathcal{C}(v)$ and assembles the survivors with a fixed renderer π, without LLM rewriting. Selection, assembly and final verification use the same snapshot and evaluation time:

$$\begin{aligned} \mathcal{C}^+(v, q) &= \{c \in \mathcal{C}(v) : \mathrm{Allow}(c, q)\}, \\ v' &= \mathrm{Render}_\pi(\mathcal{C}^+(v, q)), \\ \mathrm{ServeRebuilt}(v', q) \iff & \mathcal{C}^+(v, q) \neq \varnothing \\ & \wedge \mathrm{UniqueKeys}(\mathcal{C}^+(v, q)) \\ & \wedge \mathrm{Allow}(v', q). \end{aligned} \tag{3}$$

The rebuilt artifact depends exactly on the retained claims and is verified again. Nonempty-set and unique-key conditions exclude empty results and multiple surviving versions of one fact. The application supplies a decomposition and renderer preserving meaning and applicable restrictions, as specified in the appendix. Reconstruction preserves the content of the selected evidence.

Example. In the running example, write the plan as two claims, $\mathcal{C}(v) = \{c_{\mathrm{diet}}, c_{\mathrm{budget}}\}$, supported by Diet@1 and Budget@1, respectively. After the dietary correction, $\mathrm{Allow}(v, q) = \mathrm{false}$ but $\mathcal{C}^+(v, q) = \{c_{\mathrm{budget}}\}$: whole-artifact Gate serves neither claim, whereas reconstruction serves the budget alone. This preserves evidence for a budget question while excluding the retired preference. Adding the revised preference requires a new derivation supported by Diet@2. Figure 4 illustrates the same retention rule in a multihop question, showing the served context and recorded answer together.

4.4 EXCLUSION AND RETENTION PROPERTIES

The rules yield two conditional properties under Section 3's assumptions. The appendix provides proofs and an analysis of the relation to answer utility.

Property 1: conditional exclusion. At a fixed snapshot and evaluation time, an admitted artifact has no invalid or barred node in its recorded support: such a node would falsify a conjunct of Equation 2. Complete, accurate lineage extends this statement to actual dependencies within the serving conditions of Section 3.

Property 2: conditional retention. Let $\mathcal{C}^+$ be the claims passing the same check. If this set is nonempty, has unique logical keys, and admits a nonempty faithful rendering that passes re-verification, reconstruction retains all of it. Among policies selecting only existing individually

admissible claims, $\mathcal{C}^+$ is the greatest retained set under set inclusion. For an artifact faithfully composed of its claims, with every claim's support included in the artifact's support, let T_G and T_R denote the claim sets served by Gate and reconstruction. Under the same successful-reconstruction premises,

$$T_G \subseteq T_R = \mathcal{C}^+; \qquad T_G = \varnothing \subsetneq T_R \text{ if Gate rejects and } \mathcal{C}^+ \neq \varnothing. \tag{4}$$

Applicable restrictions remain in the retained support. Equally informed component baselines may select the same $\mathcal{C}^+$, attaining the same retention benefit over whole-artifact rejection.

From retained evidence to answers. For a fixed reader, question and renderer, let $J(T)$ be the expected answer score from context T. Section 5 compares $J(T_G)$ and $J(T_R)$ empirically, alongside exclusion and retention.

5 Evaluation: exclusion, retention, and answer utility

We ask whether verification excludes invalid derivatives (RQ1), component granularity retains unaffected information (RQ2), and retention improves downstream answers (RQ3), then measure local verification cost.

5.1 Experimental setup

Tasks and design. RQ1 uses Mem0/Graphiti lifecycle experiments and stale-derivative injections into LongMemEval-S/LoCoMo material. RQ2 uses authored route–budget tasks to examine component retention; RQ3 uses LongMemEval-V2 and selected MQuAKE-CF development questions. Qwen3.5-9B Q4 is the reader in RQ2 and RQ3. In LongMemEval-V2, Native receives the original context plus two frozen invalid items. Model settings and supplementary results are in the appendix.

Comparisons. Native serves unchecked candidates; root only updates sources without gating derivatives. Gate rejects whole artifacts, whereas reconstruction re-verifies selected components. The six-arm RQ2 comparison adds dependency-version checking and eager invalidation at the granularities shown in Table 2. Gate is paired with artifact-level eager invalidation, and reconstruction with component-level eager invalidation. This separates granularity from timing. All six methods receive the same sources, versions, dependencies, events and component boundaries; strong baselines may retain indexes and share synchronization and consumption safeguards. Regeneration's additional evidence access is specified separately.

Measures. We distinguish invalid content in the reader's input (E) from stale values used in its answer (U); lower is better for both. Retention (R) measures unaffected evidence preserved; task scores measure answer quality. RQ2's task contract requires supported answers or justified field-level abstention. Reporting groups related variants and counts shared answers as dependent observations. Unknown stale-use labels yield full-denominator bounds; malformed responses fail the contract. Definitions and statistical units are in the appendix.

5.2 RQ1: does verification exclude invalid derivatives?

Design. To isolate the effect of checking at use, we retain outdated derivatives after source changes in 25 Mem0 update/delete episodes and 25 Graphiti lifecycles. A complementary public-text intervention injects two stale artifacts into each of 2,042 LongMemEval-S/LoCoMo questions. Known-valid items measure preservation of unaffected memory. These interventions isolate exclusion and retention under specified stale-content conditions.

Serving-time checks exclude stale content while preserving valid memory. VeriMem blocks the retained stale content and preserves known-valid items, while Native and root only continue to expose the stale derivatives (Table 1). Exposure falls from 25/25 to 0/25 in each backend study and from 4,084 to zero in the public-text intervention. Checking support at use thus carries source invalidation through to the consumption of independently cached derivatives.

Study	Exposure unit	Native	Root only	VeriMem
Mem0	Episodes / 25	25	25	0
Graphiti	Lifecycles / 25	25	25	0
Public-text intervention	Injected artifacts / 4084	4084	4084	0

Table 1: Invalid-derivative exposure; lower is better. The public-text intervention injects two artifacts per question. Exposure is counted in each study's stated unit, with separate denominators.

5.3 RQ2: DOES COMPONENT GRANULARITY PRESERVE USEFUL INFORMATION?

Design. Each task asks for a route and an unaffected budget: rejecting outdated route evidence should not erase an answerable budget. The 32 inputs cover 16 related worlds from four authored families—multihop correction, partial withdrawal, new IDs and unchanged controls—under two evidence regimes. Retained-only offers no replacement route; shared-current gives every method the same separate current candidate. Overall, 16 routes require answers and 16 require abstention; all 32 budgets are answerable. Scoring uses the common candidate pool, so discarding valid evidence does not justify abstention. Identical complete requests share one Qwen response: 192 method–input comparisons use 53 distinct generations. The allocation is detailed in the appendix.

Method	Contract	Input E	Linked U	U unknown	Retained
Native	8/32	24/32	24/32	0	100%
Eager invalidation (artifact)	8/32	0/32	0/30	2	37.5%
VeriMem Gate	8/32	0/32	0/30	2	37.5%
Dependency-version (component)	32/32	0/32	0/32	0	100%
Eager invalidation (component)	32/32	0/32	0/32	0	100%
VeriMem reconstruction	32/32	0/32	0/32	0	100%

Table 2: Matched-information results, grouped as unchecked, whole-artifact and component policies. Exact requests share answers. Each component method gives 16 correct routes, 16 justified route abstentions and 32 correct budgets. Retained is mean per-instance valid retention. Linked U uses classified answers: two malformed outputs fail the contract but have unknown U, giving full-denominator bounds $[0, 2/32]$. Exposure is fully observed.

Component granularity preserves answerable information. Separating budgets from rejected route evidence explains the gain: reconstruction retains all valid components and succeeds on 32/32 inputs, compared with 37.5% retention and 8/32 success for whole-artifact Gate and eager (Table 2). It improves 24 inputs over Gate and ties eight, with no observed stale exposure or stale-value use. Native retains the budgets but also exposes and uses stale content in 24/32 inputs.

VeriMem attains the same exclusion, retention and task success as the equally informed component-level eager and version-checking baselines on all 32 inputs. This parity isolates the benefit of finer selection across maintenance timings. VeriMem realizes that benefit through checks at use; the next comparison examines answer utility, and costs are analyzed separately below. Earlier development comparisons are in the appendix.

5.4 RQ3: DOES RETAINED CONTEXT IMPROVE DOWNSTREAM ANSWERS?

Table 3 separates two questions: whether excluding invalid items helps answering (A), and whether retaining additional valid components helps beyond whole-artifact rejection (B). LongMemEval-V2 uses task accuracy; MQuAKE-CF uses our local current-alias metric. The 72 MQuAKE questions are three variants of 24 selected cases, hence related observations. We report the two studies separately.

Exclusion improves long-memory answers. On these inputs, excluding invalid items improves Qwen's accuracy by 16.7 percentage points, from 23/60 to 33/60 on the same questions: ten gains, no losses and 50 ties. Whole-item removal gives Gate and Rebuild identical complete requests, which share one answer per question in the merged column. These previously seen development questions

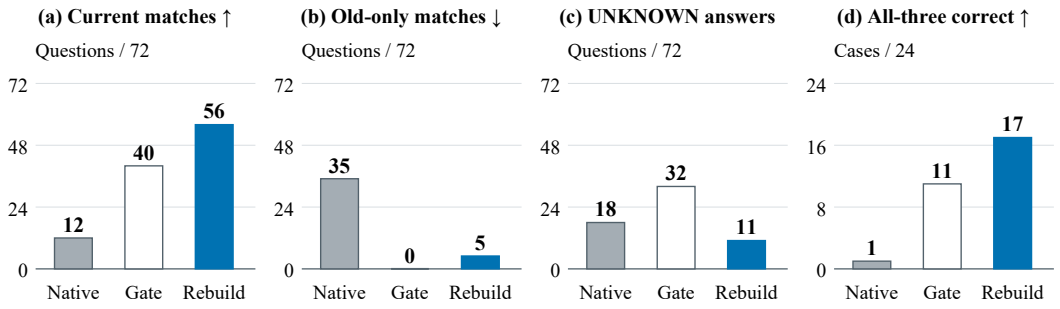

Figure 3: Answer diagnostics for the same Qwen MQuAKE-CF batch. Panels (a)–(c) count 72 questions; (d) counts cases with all three variants matching the current answer, out of 24 cases. ↑ Higher is better; ↓ lower is better. Old-only is an output-level diagnostic: an old-alias match without a current-alias match. UNKNOWN has no preferred direction: abstention can be appropriate, and it can also reduce old-only matches. Interpret the panels jointly.

A. LongMemEval-V2: invalid-content exclusion			
Reader	Unit	Native	Gate / Rebuild
Qwen	60 questions	23	33

B. MQuAKE: answer utility of retained context				
Reader	Unit	Native	Gate	Rebuild
Qwen	72 questions	12	40	56

Table 3: Correct-answer counts with Qwen3.5-9B Q4_K_M; higher is better. In A, Gate and Rebuild produce identical requests and share answers. In B, their distinct contexts are compared separately. Regeneration shares Rebuild answers in Panel B and is omitted.

isolate the answer benefit of exclusion; RQ2 and MQuAKE examine the additional role of component retention. Protocols and historical results are in the appendix.

Retention restores evidence needed for multihop answers. In contrast, the MQuAKE adaptation tests whether retaining unedited relations helps the reader answer after a source change. Reconstruction produces 56/72 current-answer matches on the selected MQuAKE cases (Zhong et al., 2023), compared with 40/72 for Gate and 12/72 for Native. The progression separates two gains: exclusion adds 28 matches over unchecked serving, and component retention adds a further 16 over whole-artifact rejection. Together with RQ2, these results connect finer information retention to improved Qwen answers in the evaluated setting.

The gain also appears at the case level (Figure 3): reconstruction raises the number of cases with all three variants correct from 11 to 17 and reduces UNKNOWN responses from 32 to 11. Alongside this higher answer coverage, old-only matches number five for reconstruction and zero for Gate.

In Figure 4, reconstruction retains the unchanged author–work link and excludes the retired language claim. The retained link completes the path from Yann Martel to the current language, changing Qwen's answer from UNKNOWN to Japanese.

5.5 Cost of admission at use

The timing choice can be evaluated by balancing avoided advance maintenance against added verification on reads. Across 12 graph shapes and nine workloads, our measurements include preparation, updates and admission. Reconstruction incurs higher cost than the component baselines because it constructs and verifies an artifact at read time, and the tested workloads show no stable total-cost saving. These measurements identify the work that must be offset by deferred maintenance; detailed accounting is in the appendix.

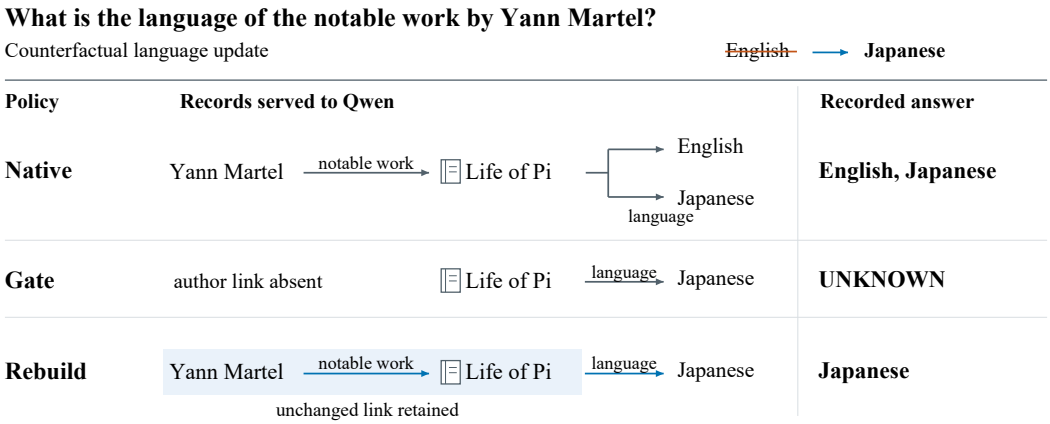

Figure 4: Component retention in a question-answering example. Under a counterfactual language update, Gate discards the old language record together with the unchanged author–work link. Rebuild retains that link and combines it with the separate current language record. Right: recorded Qwen answers. Native's mixed answer also counts as a current-alias match under the local scorer.

6 Discussion and Conclusion

VeriMem links source changes to the evidence used by a language-model reader. The studies show two complementary benefits: excluding invalid items improves the evaluated long-memory answers, while preserving unedited relations recovers multihop answers lost under whole-artifact rejection. Verification identifies reusable evidence, and relevance and reader behavior determine how that evidence contributes to an answer.

Matching the equally informed component baselines demonstrates that the retention benefit can be achieved through verification at use. The comparison attributes that benefit to finer selection and separates it from maintenance timing. VeriMem applies selection where independently cached derivatives enter model input. Deferring checks can avoid advance work on unused derivatives; frequent reads and efficient eager batching can favor earlier maintenance. The measured read-time overhead makes update frequency, dependency fan-out and repeated reads central to choosing between these strategies.

Together, the support predicate and component reconstruction provide a common serving-time rule for exclusion and selective reuse under the stated dependency and serving-state assumptions. The observed Qwen gains connect these mechanisms to improved answers in the evaluated settings. VeriMem complements memory maintenance with a verification layer at the point of consumption.

AI Use Statement

In this work, we used generative AI tools for writing refinement, partial code generation, partial figure creation, methodological design, and assistance with result interpretation. The core research idea originated with the authors, who directed the study and retained control over the experimental process. We did not use generative AI tools to originate the core research idea or to make experimental decisions without author oversight. Additionally, we used generative AI tools for grammar correction and improvements to clarity and readability. We have reviewed all AI-assisted work. The authors reviewed and revised the AI-assisted text, checked the AI-generated code for correctness, and verified that the figures accurately represented the intended content. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning*

Representations, 2024.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A characterization of data provenance. In *Database Theory – ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pp. 316–330, Berlin, Heidelberg, 2001. Springer. doi: 10.1007/3-540-44503-X_20.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. arXiv preprint arXiv:2504.19413, 2025.

Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979. doi: 10.1016/0004-3702(79)90008-0.

Ashish Gupta and Inderpal Singh Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Engineering Bulletin*, 18(2):3–18, 1995.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Belinda Z. Li, Emmy Liu, Alexis Ross, Abbas Zeitoun, Graham Neubig, and Jacob Andreas. Language modeling with editable external knowledge. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3070–3090, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.168.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747.

Andrey Mokhov, Neil Mitchell, and Simon Peyton Jones. Build systems à la carte. *Proceedings of the ACM on Programming Languages*, 2(ICFP):79:1–79:29, 2018. doi: 10.1145/3236774.

Ciyan Ouyang and Rui Hou. MemLineage: Lineage-guided enforcement for LLM agent memory. arXiv preprint arXiv:2605.14421, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. arXiv preprint arXiv:2310.08560, 2023.

Ruoming Pang, Ramon Caceres, Mike Burrows, Zhifeng Chen, Pratik Dave, Nathan Germer, Alexander Golynski, Kevin Graney, Nina Kang, Lea Kissner, Jeffrey L. Korn, Abhishek Parmar, Christina D. Richards, and Mengzhi Wang. Zanzibar: Google's consistent, global authorization system. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pp. 33–46. USENIX Association, 2019.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3586183.3606763.

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

Vikas Reddy and Sumanth Reddy Challaram. Reliable post-retrieval assembly for agent memory: Separating evidence extraction from policy execution. arXiv preprint arXiv:2606.01435v2, 2026.

Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

Igor Santos-Grueiro. Temporary authority, permanent effects: Commit-time authorization for LLM agents. arXiv preprint arXiv:2607.10487, 2026.

Fred B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, 2000.

Yi Ting Shen, Kentaroh Toyoda, and Alex Leung. Revoked but still authoritative: An empirical study of revocation enforcement in agent-memory systems. arXiv preprint arXiv:2609.08258, 2026.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 2023.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In *Proceedings of the 48th IEEE/ACM International Conference on Software Engineering*, pp. 2938–2950, New York, NY, USA, 2026a. ACM. doi: 10.1145/3744916.3764546.

Yiqi Wang, Jiaqi Liu, Jiaqi Zhang, Zhangkai Wu, Yiqun Duan, Mingkai Zheng, and Taotao Cai. PACMI: Provenance-aware cascading memory invalidation for long-term LLM agents. SSRN preprint 7480138, 2026b.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In Yisong Yue, Animesh Garg, Nanyun Peng, Fei Sha, and Rose Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 86809–86836, 2025.

Di Wu, Zixiang Ji, Asmi Kawatkar, Bryan Kwan, Jia-Chen Gu, Nanyun Peng, and Kai-Wei Chang. LongMemEval-V2: Evaluating long-term agent memory toward experienced colleagues. arXiv preprint arXiv:2605.12493, 2026.

Jinghan Xu, Yiyong Xiao, Wanru Shao, Hankai Liu, and Xinjin Li. Memory provenance laundering in LLM agents: A non-amplification firewall for persistent memory. arXiv preprint arXiv:2607.29167, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Caili Yu, Yiqi Wang, Jiaqi Zhang, Yiqun Duan, Mingkai Zheng, Zhangkai Wu, Kaize Shi, and Taotao Cai. From faulty memories to corrected actions: Dependency-guided rollback repair for memory-augmented agents. arXiv preprint arXiv:2608.10502, 2026.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. arXiv preprint arXiv:2404.13501, 2024.

Yang Zhao, Chengxiao Dai, Mengying Kou, and Yue Xiu. MemoRepair: Barrier-first cascade repair in agentic memory. arXiv preprint arXiv:2605.07242, 2026.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15686–15702, 2023. doi: 10.18653/v1/2023.emnlp-main.971.

Yan Zhou, Yue Ouyang, Kaiyang Zheng, and Suncheng Xiang. TEPA: Revoking stale memories for conflict-robust language agents. arXiv preprint arXiv:2608.07429v2, 2026.

A VERIFICATION DETAILS AND ASSUMPTIONS

A.1 SUPPORT, ADMISSION, AND RECONSTRUCTION

Let $G = (V, E)$ be a directed acyclic graph, where $(u, v) \in E$ means that v depends on u, and let $S(v) = \{v\} \cup Anc(v)$. Each record carries its lifecycle status, generation, validity interval and permitted purposes. Intervals are $I_u = [\text{valid_from}(u), \text{valid_to}(u))$, with omitted endpoints unbounded. An empty stored purpose list means unrestricted access: its effective P_u is the universe of purposes, not the mathematical empty set. A barrier records a source root, a strictly increasing mutation epoch and an active flag. Logical keys identify facts across updates, whereas immutable versions identify the values supporting a derivative. Native storage IDs, which may change on update, are not version certificates.

Adapters map backend IDs to immutable version identities and generations. Reusing an ID or returning a new one does not rebind existing derivatives to a successor; missing required records prevent admission. The implementation records invalidation before exposing a native change and quarantines unresolved effects, as detailed below.

A request supplies candidates, the graph and barrier snapshots, a purpose and an evaluation time. The adapter checks the entire supplied graph for unique identifiers, known references and acyclicity. The Gate then checks every member of $S(v)$ for active status, valid time and permitted purpose, and rejects v if any active barrier is rooted in $S(v)$. Invalid input aborts the request; applications must not fall back to unchecked candidates.

An allow certificate records the artifact, purpose, issue time, support set, support generations and barrier epoch for auditing. Each reuse requires a new Gate check and unchanged bindings. An explicit evaluation time is shared across stages. The evaluated revision samples an omitted time separately during checks and certificate issuance. A subsequent fix samples once per admission batch and has regression coverage; the reported experiments use the earlier revision.

Component reconstruction. Claims have unique immutable IDs, nonempty logical keys, positive integer versions and explicit support. Gate the claims, require at least one survivor and reject multiple surviving versions of the same logical key. Render only surviving claims using fixed code-owned framing; create an immutable artifact whose sources are exactly those claims, and Gate it again. The result binds the rendering specification and payload/output hashes. Under the same premises, its recorded support is valid. The writer supplies claim boundaries, dependencies and factual content; verification establishes the admissibility of the declared support.

A.2 ARGUMENTS FOR THE STATED PROPERTIES

The following arguments use one validated graph, barrier snapshot, purpose and evaluation time across both verification stages. Constructing v' extends the graph by its fresh node and incoming support edges, preserving existing records, dependencies and lifecycle state.

Exclusion. Suppose an admitted v has a node $u \in S(v)$ that is inactive, time-invalid, purpose-incompatible or barred. One conjunct of Equation 2 is then false, contradicting admission. This establishes exclusion for recorded support. Complete, accurate attribution extends the result to actual dependencies; faithful rendering and mandatory admission checks carry it through to model input.

Retention and compositional validity. Write $C = \mathcal{C}(v)$ and $C^+ = \mathcal{C}^+(v, q)$. Any permitted selector returns $T \subseteq C$ whose every member passes the Gate; hence $T \subseteq C^+$. If C^+ is nonempty with unique logical keys, and faithful rendering produces nonempty output and re-verification succeeds, reconstruction returns exactly C^+. Thus it is the greatest retained set within this class. Duplicate logical keys cause rejection.

With the rebuilt artifact's direct sources set to the retained claims,

$$S(v') = \{v'\} \cup \bigcup_{c \in C^+} S(c). \tag{5}$$

All nodes in each $S(c)$ pass the same snapshot check. Final verification covers both v' and its complete closure. Faithful framing preserves claim meaning and introduces only supported assertions.

Component support must include applicable artifact-wide restrictions, or the output must enforce them separately. The application supplies a decomposition that preserves these restrictions and any contextual dependencies between claims.

For the whole-artifact comparison, additionally require $\bigcup_{c \in C} S(c) \subseteq S(v)$ and that serving v faithfully exposes exactly C. If Gate accepts v, support closure and the shared predicate imply that every c passes, so $T_G = C = C^+ = T_R$. If it rejects, $T_G = \varnothing$; successful reconstruction returns C^+, giving strict inclusion when nonempty. This ordering applies under the successful-reconstruction premises above, with restrictions preserved through rebinding. Equally informed component baselines can select the same C^+: the property identifies the retention benefit of component granularity across maintenance strategies.

Support validity and answer utility. Let T_0 contain a supported answer-bearing path, and let d be a supported fact about a different entity. A deterministic reader that follows the path on T_0 but answers using d on $T_0 \cup \{d\}$ is consistent with all support assumptions. For binary correctness it has $J(T_0) = 1$ and $J(T_0 \cup \{d\}) = 0$. Conversely, an added claim that supplies a missing answer can improve utility. These logical constructions separate support validity from reader behavior: the same retention rule can improve or reduce answer quality. The paired reader comparisons measure which effect occurs in the evaluated settings.

Computational scope. Checking a candidate traverses its support closure and tests active barriers. The current implementation sorts adjacency lists and materializes certificate bindings; claim-wise verification can revisit shared ancestors before the rebuilt artifact is checked again. Graph construction, rendering and persistence add separate work. Section 5 measures these implementation costs, complementing the exclusion and retention properties.

A.3 Integration boundary

Source updates must make affected support ineligible before a reader can admit it. The implementation durably accepts an intent and projects pending intents into rejection; for a change with a successor, it then orders

$$\mathrm{PersistBarrier}(e{+}1) \prec \mathrm{MutateNative} \prec \mathrm{BindSuccessor}\,. \tag{6}$$

A deletion omits successor binding. Unresolved effects remain quarantined until backend-specific readback establishes the native outcome. Crash recovery and cross-store coordination remain backend integration responsibilities; the ordering rule specifies when dependent content becomes ineligible.

The serving snapshot must remain applicable at consumption. For support S, consumption at t_u additionally requires

$$t_u < \tau_{\min}(S), \qquad \tau_{\min}(S) = \min_{s \in S} \mathrm{expires}(s), \tag{7}$$

with unbounded expiry $+\infty$. Admission and consumption must both satisfy the validity conditions. Implementation probes found unresolved recovery and post-check expiry issues. Figure 5 identifies the required writer coordination and consumption-time expiry check; these remain integration requirements.

A.4 Scope of the contract

Dependency quality determines how faithfully the recorded graph represents actual support. Omitting a retired address source, or replacing it with an unrelated valid source, can allow the old address to pass. Attaching every input to every claim can instead invalidate unchanged opening hours when only the address is withdrawn. Structural validation detects unknown IDs; the writer is responsible for semantically complete and accurate bindings. VeriMem enforces the resulting support graph at admission.

Memory admission and action enforcement protect distinct stages. AgentSpec enforces runtime policies, while MemLineage and PPMF constrain provenance-based action or argument authority (Wang et al., 2026a; Ouyang & Hou, 2026; Xu et al., 2026). VeriMem's evidence checks complement these action-level controls. Table 4 summarizes their overlapping contracts.

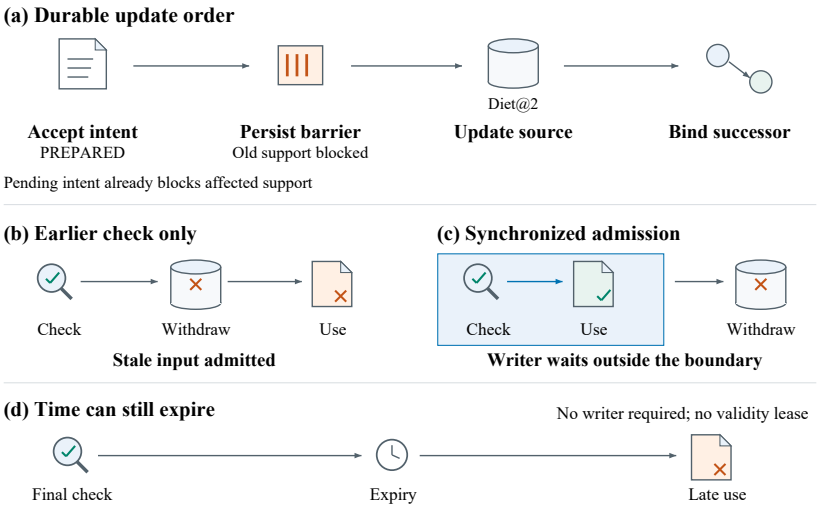

Figure 5: Integration requirements. (a) Record accepted invalidation before exposing a native change. (b–c) Coordinate admission with writers to preserve the checked state; the same coordination is available to strong baselines. (d) Check expiry at consumption even with writer serialization. This schematic separates update ordering, writer coordination and time validity.

Work	Protected object and enforcement boundary
MemoRepair	Stored descendants: withdrawal before repair and validated, predecessor-closed republication (Zhao et al., 2026).
PACMI	Memory validity: propagation through typed dependencies, with state-aware retrieval and stale-premise detection (Wang et al., 2026b).
Read-time guard	Returned records: withholding revoked facts or facts conflicting with their replacements (Shen et al., 2026).
Zanzibar	Authorization decisions consistent with access-policy and object changes (Pang et al., 2019).
CommitGuard	Durable actions: fresh authority and effect binding at a coupled check-and-commit boundary (Santos-Grueiro, 2026).
Rollback repair	Memory and traces: dependency-guided deactivation and selective replay (Yu et al., 2026).
VeriMem	Retrieved derivatives: versioned support checks and optional reconstruction before context construction.

Table 4: Conceptual comparison of selected memory and enforcement contracts. Memory admission and action-commit authorization address complementary stages; experimental results for our policy baselines are reported separately.

B Experimental Details

B.1 Comparison Design and Model Settings

Each comparison fixes the task, source state, candidates and scoring rule before observing condition outcomes. Matched component methods receive the same sources, versions, dependencies and events; dependency-version checking and eager invalidation may retain fingerprints or reverse indexes. Shared coordinator and consumer protections are available to all methods. Regeneration may receive additional current evidence and is not always information-matched. Tables 5 and 6 summarize study units and reader settings, respectively.

The 32-input comparison uses four Qwen workers and 4,096 tokens per slot, without automatic retries. Its 53 distinct requests pass capacity checks before generation; the batch completes without failures or retries. The current LongMemEval-V2 Qwen reader uses top-p 1, thinking disabled, top-k 20 and one 131,072-token slot. Complete request and scoring details, including capped outputs, are given

Study	Unit and coverage
RQ1: backend lifecycle	25 Mem0 and 25 Graphiti episodes.
RQ1: public-text intervention	2,042 LongMemEval-S/LoCoMo questions; two injected derivatives each.
RQ2: matched information	32 inputs, 16 related worlds, four templates; 192 logical cells and 53 distinct generations.
RQ3: LongMemEval-V2	60 previously seen questions (30 Web, 30 Enterprise); stale-item injection.
RQ3: MQuAKE	24 selected source-related groups, three question variants each.

Table 5: Units of the main and supplementary comparisons. Related variants and shared requests do not increase the independent sample size. Historical studies are reported separately in the supplementary results.

Current reader comparison	Model	T	Output cap
RQ2: matched information	Qwen3.5-9B Q4_K_M	0	256
RQ3: LongMemEval-V2	Qwen3.5-9B Q4_K_M	0	8,192
RQ3: MQuAKE	Qwen3.5-9B Q4_K_M	0	64

Table 6: Reader settings for the main and supplementary comparisons. Output caps are tokens, not observed response lengths.

in the supplementary results. Response sharing requires identical complete serialized requests, not merely the same model or similar text; tied shared answers are not independent replications.

Historical studies. The 20-item LoCoMo category-5 study measures following of the public adversarial answer, not overall LoCoMo accuracy. The historical Qwen 60-question scores are not results of the current matched batch.

B.2 Metrics and statistical units

Table 7 defines the endpoints used to distinguish exclusion, retention, answer utility and local cost.

Endpoint	Observation
Input exposure E	Identified invalid content in serialized model input.
Linked stale use U	Attributable old-value use; ambiguous labels stay unknown.
Valid retention R	Known-valid components retained, not answer correctness.
Task utility	Fixed route–budget contracts (RQ2); official deterministic functions and frozen judge rubric on LongMemEval-V2 inputs with stale-item injection; local current-alias matches for MQuAKE, not its official score.
Local cost	Scoped policy-query and graph-preparation measurements.

Table 7: Exclusion, retention and answer quality are separate endpoints. Applying a benchmark's scorer to modified inputs does not turn the result into an unmodified benchmark evaluation.

Current main-text reader comparisons use descriptive counts and paired gains, losses and ties. MQuAKE question variants share 24 cases, and the 32-input study shares 16 worlds. For historical analyses, a paired sign test concerns directions of non-tied outcomes; a bootstrap interval describes uncertainty with related observations resampled together. The two-sided $p = 0.1185$ and interval belong to historical Qwen 24/60 versus 31/60, while the current 23/60 versus 33/60 batch is reported descriptively. Each batch retains its own statistical analysis. Structural counts and local timings are descriptive.

For N inputs, a observed stale uses and m unknown labels,

$$\widehat{U}_{\text{obs}} = \frac{a}{N-m} \quad (N > m), \qquad U_{\text{full}} \in \left[\frac{a}{N}, \frac{a+m}{N}\right]. \tag{8}$$

These are missing-label bounds, not confidence intervals. Unknown labels remain unknown, and malformed answers fail the task contract even when U cannot be classified. Interrupted prefixes are not combined with later runs.

Reproducibility scope. Input rosters, configurations, responses and scores are retained per experiment. Offline rescoring verifies recorded evidence, not independent model replication. The manuscript package contains sources, figures and references, not private raw trajectories or model weights. Full development records remain in the research archive. This manuscript reports Qwen as the answer-generating reader; additional-model development runs are outside its reported scope and remain archived. Results refer to their frozen revisions, including the clock handling described in Appendix A.

C SUPPLEMENTARY RESULTS

C.1 MEMORY AND RETENTION STUDIES

The backend comparisons deliberately retain derivatives with recorded support. Mem0 2.0.18 uses local Qdrant and an instrumented writer; Graphiti uses local Kuzu and deterministic embeddings. These studies test enforcement, not natural stale-memory prevalence. The public-text intervention injects two derivatives per question: no gate and root-only each expose 4,084 artifacts, whereas VeriMem exposes none and retains all known-valid items. Both backend studies also attain full certificate coverage for the recorded admission decisions; this is an implementation diagnostic, separate from the exposure and retention measures. Table 8 summarizes the public-text intervention and historical reader comparisons.

Table 8: Public-data comparisons. Lower is better for exposure and adversarial following; higher is better for semantic accuracy. Structural counts have no confirmatory significance test.

Protocol	n	Metric	No gate	VeriMem	Δ	p
Public-text intervention	2042	Stale-artifact exposure	4084	0	-4084	–
LoCoMo category 5	20	Adversarial exact following	12	0	-12	–
LongMemEval-V2 / Qwen (historical)	60	official accuracy	.400	.517	+.1167	.1185

The historical Qwen 60-question comparison has 11 improvements, four losses and 45 ties, with two-sided $p = 0.1185$ and bootstrap interval $[0, 0.2333]$. The observed difference is positive, with statistical uncertainty that includes zero.

In a separate Mem0 withdrawal study, 20 deletions use copies of five known snapshots, with five unchanged controls. Native exposes 20 withdrawn components; Gate, reconstruction and regeneration expose none. Of 60 valid components, Gate retains zero and the other methods retain all. Reconstruction and regeneration are byte-identical; no downstream consumer is called. In 12 authored tasks, Gate succeeds in 0/12, while native, reconstruction and oracle each succeed in 12/12. Native follows an explicit correction despite stale exposure in all 12 tasks. Gate retains one third of valid claims, compared with full retention in the other arms. These results connect finer retention to answer availability, while Native's success illustrates how a reader can also follow an explicit correction despite stale exposure.

For the matched-information comparison in the main text, four families each contain four worlds under two evidence regimes. Retained-only supports four route answers; shared-current supports 12. Each component method's 32/32 score comprises 16 grounded routes, 16 justified route abstentions and 32 correct budgets. The three methods share responses for identical requests. Whole-artifact methods have zero linked stale uses among 30 classified outputs and two unknowns, giving full-denominator bounds of $[0, 2/32]$. Eight development inputs are excluded; a shared empty-memory request is generated anew.

C.2 Downstream comparisons

Matched LongMemEval-V2 reading. The 25 September Qwen comparison reuses 60 previously seen questions, 30 per domain. Native contains the original context plus two frozen invalid items. Gate and whole-item reconstruction run separately, but after common boundary whitespace normalization yield identical complete requests. Their 60 pairs share answers and 13 judge results, giving 180 logical cells from 120 reader and 26 judge calls. Text content, images and ordering are preserved. Qwen3.5-9B Q4_K_M uses temperature 0, top-p 1, thinking disabled, an 8,192-token output cap, top-k 20 and one 131,072-token slot, without retries. All requests complete, with no input truncation or missing scores. Seven Qwen outputs reach the output cap (three Native, four shared Gate/Rebuild); all are retained and scored. Official deterministic functions score 141 logical cells; the frozen official LLM-judge rubric and parser score the remaining 39 using `deepseek-flash` as a judge, not as the answer-generating reader. Raw judge responses are replayed offline. Qwen Native scores 9/30 Web and 14/30 Enterprise; shared Gate/Rebuild scores 15/30 and 18/30. The completed Qwen batch uses fresh answers and judgments throughout, not prefixes from the cancelled batch or historical scores. These are development comparisons of exclusion, not independent reconstruction gains or new holdouts.

Long-memory retrieval. A separate eight-question query-conditioned pilot has five usable inputs, two undefined recipes and one missing snapshot. Across 20 observed answers, all arms tie at 1/5; 12 cells remain unobserved.

C.3 MQuAKE selection and answer analysis

MQuAKE-CF provides 9,218 inventoried cases, of which 3,160 pass a fixed structural filter. Grouping by shared source–relation keys or question hashes yields 819 groups. Before outcomes, seeded SHA-256 ranking selects one representative from each of 24 groups; 16 related cases are quarantined. The seed is `verimem-mquake-supplement-20260915-v1`. Selected IDs are 42, 102, 507, 911, 927, 1603, 1620, 1720, 1747, 2653, 2685, 2691, 2856, 2865, 3162, 3461, 3463, 3591, 3606, 3976, 4172, 4366, 4813 and 4866. All have two hops and three question variants. We compile 31 edits into memory lifecycles while retaining unedited path facts. Mem0 preparation precedes reading; aliases reduce each four-condition grid to 216 physical requests. This selected development subset serves as a multihop stress test; its scores describe the subset rather than the full official benchmark.

Native/Gate/reconstruction/regeneration answer matches are 12/40/56/56 of 72 for Qwen. Regeneration shares reconstruction's inputs and answers. Earlier same-case Qwen projection reaches 63/72 (eight gains, one loss), but is development evidence rather than independent confirmation.

Evidence scope. The studies connect source invalidation, component retention and Qwen answering on the specified development samples. Results are interpreted within each study and reader, grouping related variants and shared responses by source. This evidence concerns memory exposure, retention and answering, rather than general tool execution or agent safety. Engineering diagnostics are retained in the research archive.

D Local verification cost

D.1 Maintenance and read-time work

For an observation interval, compare policies with the same output granularity and validity checks. Let F denote the affected dependency footprint and Q the number of candidate reads, including repetitions. Let $P(F)$ be eager's preparation and update-maintenance cost, B VeriMem's corresponding setup and source-barrier cost, and $\bar{r}_E, \bar{r}_V$ their mean per-read costs. Including graph validation, reconstruction and synchronization where required, the policy-work accounting gives

$$\begin{aligned} C_E = P(F) + Q\bar{r}_E, \qquad C_V = B + Q\bar{r}_V, \\ C_V < C_E \iff P(F) - B > Q(\bar{r}_V - \bar{r}_E). \end{aligned} \tag{9}$$

The inequality identifies when avoided advance maintenance can offset verification on reads. Large affected footprints and few reads can favor deferred checking; frequent reuse can favor eager

maintenance. Batching and index reuse affect both sides of this workload-dependent trade-off. The measurements below evaluate cost separately from retention.

D.2 POLICY-QUERY MEASUREMENT

Cold/query measurements use 432 fresh policy objects, with two warmup and seven measured queries per state; objects persist across states within each cold repetition. Reconstruction costs more than the component baselines: version checking reuses support fingerprints and eager materializes reverse dependencies, whereas reconstruction builds graphs, checks claims, renders and hashes an artifact, then validates it. Timings describe this implementation, not theoretical lower bounds. They exclude journals, locks, native operations and consumer persistence; nested intervals are non-additive.

D.3 PREPARED-GRAPH COMPARISON

Table 9 compares 12 graph shapes (three depths, two fan-outs, two background sizes), nine workloads and five policies. One warmup and three measured rounds produce 2,160 executions and 29,520 queries; 1,620 executions and 22,140 queries enter analysis. Each cell starts a fresh policy and charges validation, preparation and supplied-successor publication. The optional prepared variant caches eligible graph structure, never permission, and preserves dynamic checks and final validation. Input generation, scoring and JSON persistence are excluded. Paired analysis was specified in advance; the median summary was implemented after execution.

Workload	Version	Eager	Gate	Rebuild	Prepared	Paired ratio
`steady1`	0.590	0.568	0.741	2.027	2.715	1.278
`steady20`	0.815	0.756	4.303	24.724	25.382	1.008
`pending20`	0.783	0.756	4.247	25.565	24.534	1.041
`retired1`	0.667	0.599	0.801	2.007	2.459	1.301
`retired20`	0.838	0.758	4.137	24.622	24.576	1.023
`append1`	1.278	1.192	1.428	3.146	4.754	1.443
`append20`	1.556	1.464	5.226	39.555	38.749	0.983
`abort_pending20`	0.833	0.754	4.341	24.326	24.102	1.012
`expiry20`	0.811	0.764	3.987	23.483	23.771	1.023

Table 9: Total kernel cost (ms), including preparation, updates and queries: median of 12 graph-specific medians over three measured rounds. The last column summarizes the 12 paired Prepared/Rebuild ratios, rather than dividing aggregate times. Workloads ending in 1 use one query; others use twenty. These local measurements describe repeated workloads, not independent tasks or isolated-host throughput.

Version checking, eager and prepared reconstruction match ordinary reconstruction on 13,284 paired outputs. Each arm excludes invalid content in 4,428 queries and returns supplied successors in 756/756 append queries. Gate omits 3,312/13,518 eligible claim-observations; component methods retain all. Counts include repeated reads. Prepared reconstruction shows no stable total-cost improvement (workload-level paired ratios to the two strong baselines: 3.59–35.57). These kernel measurements exclude RAM, isolated-host throughput and end-to-end model costs.