Beyond Retrieval: Deterministic Memory Governance with Pairwise Constraints and Certified Repair for Long-Term Language Agents

Panayiotis Christou[1] **Miguel Palma**[1] **Zefan Du**[1]
Kaixun Hua[2] **Ying Mao**[1]
[1]Fordham University, Bronx, NY 10458, USA
[2]University of South Florida, Tampa, FL 33620, USA
{pc33,mip2,zdu19,ymao41}@fordham.edu
khua@usf.edu

Abstract

Long-term memory significantly enhances language-model agents but introduces a critical failure mode: retrieval surfaces relevant yet stale, deleted, or mutually inconsistent facts. A query about a current address may retrieve the correct address alongside deleted ones. The answer is present, but the memory set is unsafe. We reframe agent memory as a governed object-selection problem and present MemGov, a substrate-agnostic middleware layer between existing retrieval pipelines and agent context. MemGov wraps retrieved evidence as explicit memory objects, detects must-link and cannot-link constraints, and applies deterministic constraint-aware repair, returning either a certified safe memory set or an auditable obstruction certificate rather than silently passing unsafe combinations. Evaluations on controlled governance scenarios and MemoryAgentBench-style end-to-end tasks, deliberately separating retrieval quality from governance decisions, demonstrate that while raw retrieval recovers answer-bearing evidence successfully, it simultaneously returns detected unsafe memory combinations in 47–51% of cases. The same retrieval substrates governed by MemGov reduce unsafe merges to 0% while preserving answer utility within about 2–4 percentage points, with fully auditable reasoning for all repair and refusal decisions. Additional diagnostics assess eligibility, temporal constraints, and paired answer retention.

1 Introduction

Language-model agents are increasingly deployed with long-term memory: stored facts, user preferences, task histories, and episodic observations that persist across interactions (Zheng et al., 2026). This capability makes agents substantially more useful. An assistant that remembers a user recently moved, that a project deadline changed, or that two earlier instructions refer to the same task can provide far more coherent and personalized assistance than one that starts fresh at every session. As agentic memory systems grow more capable (Packer et al., 2023; Xu et al., 2025; Hu et al., 2025), however, a fundamental tension emerges: the same persistence that makes memory valuable also makes it dangerous (Chen, 2026).

The dominant paradigm for memory access is retrieval: ranking stored memories by relevance and passing the top candidates to the agent. This treats memory governance as a ranking problem, assuming the right answer follows relevance. But relevance and safety are orthogonal. Consider a user who recently moved: their memory store contains a current address, a superseded address, and a deleted address. All three are topically relevant to a query about their residence. A relevance-only retriever may return all three, which standard utility metrics reward because the answer is present. Yet the returned set is unsafe, mixing deleted, stale, and contradictory facts. Better ranking cannot fix this: even a perfect ranker placing the correct address first still passes inadmissible memories forward if it cannot reason about their mutual compatibility.

This failure mode extends beyond addresses. An agent might retrieve a memory belonging to a different user, use a private memory for an unauthorized purpose, or retrieve individually plausible but jointly incompatible facts (e.g., conflicting project deadlines or contradictory instructions). In each case, retrieval may find the right answer while returning a memory set that should never be merged. Relevance ranking alone does not make this admissibility determination.

We argue this gap requires treating agent memory as a **governed object-selection problem**. Retrieval determines which memories are candidates and governance determines which are admissible, individually and in combination. A memory can be highly relevant yet ineligible (e.g., deleted or unauthorized), or two eligible memories can be mutually unsafe (e.g., expressing contradictory claims). Nei-

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 5501–5518
November 6-10, 2026 ©2026 Association for Computational Linguistics

ther decision can be reduced to a similarity score.

We propose MemGov, a **substrate-agnostic middleware layer** that sits between existing retrieval or extraction pipelines and the final agent context. Rather than replacing retrieval, MemGov governs its output. The **key innovation** is a deterministic, certificate-backed repair mechanism that creates a governed boundary after retrieval: relevant evidence is checked for eligibility and pairwise consistency before it becomes agent context. Retrieved evidence is wrapped as explicit memory objects carrying governance and provenance metadata. Pairwise relationships between objects are detected and typed as *must-link* constraints, which require two memories to be used together, or *cannot-link* constraints, which prohibit two memories from appearing together. A deterministic repair layer then either produces a certified safe memory set or issues an auditable obstruction certificate explaining why no safe assignment exists. The agent receives a memory set that has passed eligibility and consistency checks, or an explicit refusal explaining the incompatible memories.

We evaluate MemGov on controlled governance scenarios with MemoryAgentBench(Hu et al., 2026), separating retrieval quality from governance quality. Raw retrieval recovers answer-bearing evidence at useful rates but returns cannot-linked memory pairs in 47–51% of evaluated cases. MemGov has zero observed unsafe merges over the detected constraints with negligible overhead and minimal impact on answer utility. MemGov makes the following major contributions:

- It frames long-term agent memory as **governed object selection**, where authorization, deletion, staleness, and pairwise consistency must be evaluated alongside relevance.
- It introduces MemGov, a **substrate-agnostic middleware layer** that wraps retrieved evidence as memory objects, enforces typed pairwise constraints, and emits auditable repair or refusal certificates.
- It defines an evaluation protocol that separates retrieval coverage from governance correctness, measuring both answer utility and unsafe merge rate.
- It demonstrates that MemGov eliminates unsafe merges entirely (**0% unsafe rate**) while preserving answer utility within 2–4% of raw systems with negligible runtime overhead ($<0.05\%$).

2 Background and Related Work

Persistent agent memory improves continuity across interactions, but it also turns stored context into a safety surface. Generative agents, MemoryBank, MemGPT, Reflexion, Voyager, and newer agentic-memory systems demonstrate the value of long-lived user preferences, task histories, observations, and decisions (Park et al., 2023; Zhong et al., 2024; Packer et al., 2023; Shinn et al., 2023; Wang et al., 2024, 2023; Xu et al., 2025; Hu et al., 2025; Saleh et al., 2025; Kang et al., 2025; Sarin et al., 2025). These systems differ in how they summarize experience, retrieve prior episodes, or organize memory over time, but share a common pressure point: once memories persist, the system must decide not only what is relevant, but also what is authorized, current, and mutually consistent.

Retrieval-augmented generation, dense retrieval, late interaction, query expansion, self-correcting retrieval, modern embeddings, and graph retrieval provide increasingly strong ways to find relevant context (Lewis et al., 2020; Karpukhin et al., 2020; Khattab and Zaharia, 2020; Gao et al., 2023; Asai et al., 2024; Jiang et al., 2023; Yan et al., 2024; Chen et al., 2024; Edge et al., 2024; Guo et al., 2025; Gutiérrez et al., 2024). MemGov treats these methods as substrates rather than competitors: stronger retrieval brings more answer-bearing evidence into the working set, but relevance alone does not determine whether a memory is authorized, current, non-deleted, or safe to combine with other retrieved memories.

SSGM filters reads, checks consistency before consolidation, and reconciles against an episodic log (Lam et al., 2026). Collaborative Memory combines provenance-aware private/shared fragments with changing read/write permissions (Rezazadeh et al., 2025). Zep invalidates temporally overlapping contradictory edges while retaining history (Rasmussen et al., 2025). The temporal-rag implementation filters expired documents and reranks by freshness and validity (Emmimal, n.d.). MemGov checks eligibility and interacting must-link/cannot-link constraints after retrieval, with deterministic repair and locally checkable certificates (Table 3).

MemGov draws on three complementary bodies of work beyond retrieval. Natural language inference (NLI) and fact verification estimate compati-

bility and contradiction between statements (Bowman et al., 2015; Williams et al., 2018; Nie et al., 2020; Thorne et al., 2018; Devlin et al., 2019; Liu et al., 2019; He et al., 2021). These detectors supply relationships, while the governance layer decides which memory set satisfies them. Privacy work on contextual integrity, erasure, and machine unlearning motivates explicit eligibility checks (Nissenbaum, 2004; European Parliament and Council of the European Union, 2016; Bourtoule et al., 2021; Xu et al., 2023). Pairwise must-link and cannot-link constraints are well studied in constrained clustering (Wagstaff et al., 2001; Basu et al., 2004, 2008), and certified subset-repair work repairs bounded working sets with checkable certificates (Chumpitaz-Flores et al., 2026). Our contribution combines these responsibilities at the agent memory boundary: retrieval proposes candidates, detectors provide typed relationships, eligibility checks exclude inadmissible objects, and constrained repair returns a verified set or an auditable refusal.

3 MemGov Solution Design

Retrieval proposes relevant memories. MemGov checks which candidates are eligible and which can safely be used together. It represents candidate evidence as governed memory objects and returns either a verified memory set or an auditable refusal.

3.1 Problem Setting

The certificate applies to the bounded working set and the supplied eligibility predicates and pairwise constraints. It verifies their satisfaction before memory reaches the agent context.

We study an agent with a long-term memory store and a query q. A standard retriever ranks memory text by relevance. MemGov instead treats each memory as a governed object: a text span plus extracted factual content, provenance, ownership, purpose, sensitivity, trust, lifecycle state, and timestamp. This object boundary matters because the same memory can be relevant to a query and still be unsafe to use. For example, a travel-planning agent may retrieve a deleted address, a stale address, and the current address because all three mention the user's home. Relevance alone does not distinguish the memory that should be used from those that should be excluded or separated.

Eligibility constraints. Eligibility constraints handle single-memory admissibility. They decide whether a memory object m may be used in a governance context γ, where γ encodes the requesting user, capability, purpose, and sensitivity budget. We write this as $E(m, \gamma) \in \{0, 1\}$. Deleted memories, cross-user memories, over-sensitive memories, or memories outside the current purpose are ineligible even if they are highly ranked by the retrieval substrate.

Consistency constraints. While eligibility filters individual memories, consistency constraints handle relationships between eligible memories via pairwise constraints. A **must-link** constraint requires two memory objects to be used together, such as duplicate evidence for the same current address. A **cannot-link** constraint prohibits two memory objects from appearing together, such as a stale address and a current address for the same user, or two incompatible citizenship claims for the same person. NLI-style detectors propose these relationships, but the repair layer decides how they affect the returned memory set. For a returned set R, the central consistency condition is:

$$(m_i, m_j) \in \mathcal{D} \Rightarrow \{m_i, m_j\} \nsubseteq R,$$

where $\mathcal{D}$ is the set of cannot-link pairs. We define a **safe memory set** as a returned set R that satisfies all eligibility constraints and contains zero pairwise constraint violations: every must-link pair is either fully included or fully excluded, and every cannot-link pair has at most one member in R. Thus an unsafe merge is not merely an incorrect answer: it is a structural violation that combines objects the governance layer knows should be separated.

Governed objective. The goal is to return a **safe memory set** that is useful for the query while satisfying all eligibility and constraints. The retrieval score $s(q, m)$ estimates candidate utility. MemGov constrains which high-scoring candidates are jointly admissible. If no safe assignment exists inside the retrieved working set, the correct behavior is an auditable refusal rather than a silent merge of inconsistent memory. Appendix B gives the expanded object notation and constrained selection.

An example from MemoryAgentBench. Figure 1 illustrates a representative case study from the MemoryAgentBench dataset (Hu et al., 2026). When a user queries the birthplace of Thomas Arne, the retrieval phase employs a reranker to score and

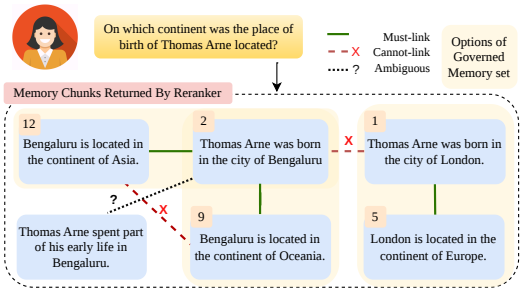


Figure 1: A case from MemoryAgentBench Dataset (Hu et al., 2026). MemGov resolves memory retrieval contradictions via pairwise constraints. Mutually exclusive facts (Rank-1 vs. Rank-2) are isolated using Cannot-Link relations, while reinforcing evidence (Rank-1 and Rank-5) is bundled with Must-Link relations to produce a conflict-free memory set.

order the relevant memory chunks. However, the top-ranked retrieved results surface contradictory evidence along the same reasoning chain. For instance, the top-ranked chunk (Rank-1) states that "Thomas Arne was born in the city of London," whereas the second-ranked chunk (Rank-2) asserts that "Thomas Arne was born in the city of Bengaluru." Consequently, to satisfy strict consistency constraints, these two mutually exclusive memory chunks should not coexist within a single retrieved set. Under our MemGov problem formulation, this conflict is explicitly modeled as a Cannot-Link relationship. Conversely, Rank-1 shares a Must-Link relationship with Rank-5 ("London is located in the continent of Europe"), as both chunks provide complementary, logically consistent evidence. Meanwhile, the relationship regarding "Thomas Arne spent part of his early life in Bengaluru" remains ambiguous. Leveraging this structured constraint analysis, MemGov effectively filters out conflicting information and returns only safe memory sets that strictly adhere to all Must-Link and Cannot-Link constraints.

3.2 MemGov System Overview

Figure 2 places MemGov between evidence discovery and agent context. Lexical, dense, hybrid, or graph-expanded retrieval, reranking, and language-model extraction provide candidate evidence. The governance layer checks object eligibility and pairwise consistency before return. Section 4 specifies the experimental retrieval and model stack.

The workflow has five steps. First, the memory substrate proposes a bounded working set for the query. Second, MemGov constructs memory objects from that evidence by attaching stable identity, provenance, ownership, purpose, sensitivity, lifecycle state, timestamp, and extracted factual atoms. Third, eligibility constraints remove individually ineligible memories, such as deleted or out-of-purpose records. Fourth, relationship detection proposes must-link, cannot-link, or ambiguous relationships among the remaining objects. Finally, the repair layer either returns a consistent subset, separates incompatible objects, or refuses if the working set cannot satisfy the hard constraints.

The output pairs a checked memory set with a certificate recording the working set, constraints, repair decision, and verification status. This makes admissibility explicit before candidate evidence reaches the agent. Appendix A expands the worked examples.

3.3 Memory Construction and Constraint Detection

Object construction. MemGov does not repair raw paragraphs. It first converts retrieved evidence into memory objects that can be checked. A language-model extractor reads selected chunks and produces grounded factual atoms, such as subject, relation, value, and evidence span. Each atom is then wrapped with provenance and governance metadata: source, owner, purpose, sensitivity, trust, and lifecycle state. In the address example, a sentence about the user's current address becomes one object, while a deleted address becomes a separate object whose state makes it constraint-ineligible.

Constraint detection. After object construction, MemGov generates candidate pairs only among memories that are structurally comparable. Two objects become comparable when their atoms share a normalized subject and relation, such as two address claims about the same user or two citizenship claims about the same person. A detector then labels the pair as compatible, contradictory, or ambiguous. Compatible pairs can produce must-link constraints. Contradictory pairs can produce cannot-link constraints. Ambiguous pairs are recorded as telemetry but are not promoted to hard constraints.

This separation makes failures interpretable. If no comparable pair is generated, the detector never had an opportunity to label it. If a pair remains ambiguous, the uncertainty belongs to the detector. If a hard constraint is generated and the governed

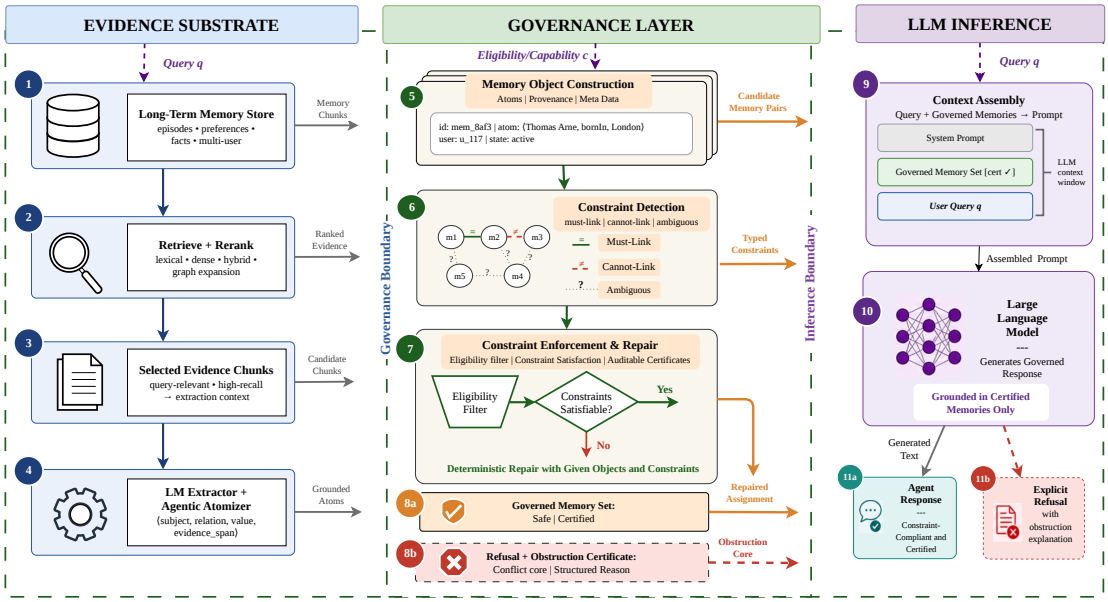


Figure 2: MemGov as middleware for agent memory. Existing memory substrates retrieve, rerank, expand, or extract candidate evidence. The governance layer converts that evidence into constraint-bearing memory objects, checks eligibility and consistency, and returns either a safe memory set or a certificate-backed refusal before memory reaches the agent context.

output still violates it, the failure belongs to the governance layer. Figure 5 in the appendix shows the full object-construction and governance flow, and Appendix C gives the extraction and detector telemetry used to audit this stage.

3.4 Eligibility Enforcement and Constraint Repair

Eligibility filtering removes memories that are individually ineligible, such as deleted, cross-user, or over-sensitive objects. But two memories can both be individually eligible and still be unsafe together. A stale address and a current address may both be active memories. Two extracted claims may both be grounded in evidence. Two answer-bearing chunks may both be highly ranked. Repair is needed for this second failure mode: deciding whether the eligible memories proposed by retrieval can be returned as one consistent set.

Deterministic repair. After eligibility filtering, the repair layer receives a bounded working set of active memory objects and typed constraints. Must-link constraints are closed with disjoint set union, so each connected component is treated as a unit. Cannot-link constraints are then checked against those components. If a cannot-link falls inside a must-link component, the constraints are obstructed and the system refuses. Otherwise, the repair routine assigns the working-set objects to consistent groups under fixed solver settings and then verifies the result by checking every hard cannot-link in the working set.

The procedure is deterministic for a fixed retrieved set, eligibility state, and constraint graph: the same inputs produce the same must-link components, the same obstruction check, the same repair assignment, and the same local verification result. The repaired assignment is not the answer shown to the language model. It is the internal structure used to decide which memories may be returned together.

Tractability and auditability. General must-link/cannot-link partitioning can be hard, but MemGov does not solve an unrestricted global clustering problem over the full memory store. It repairs only the small working set produced by retrieval, reranking, extraction, and eligibility filtering. In the main experiments this working set is the top-25 memory context. The safety claim also does not require global optimality: immediate contradictions are detected exactly during constraint compilation, and the returned set is accepted only if local verification confirms that the hard relationships are satisfied.

Certificates and refusals. Satisfiable cases produce a repaired memory set and a certificate recording the working set, constraint counts, assignment,

residual violations, and local verification status. Obstructed cases produce a refusal certificate identifying the conflicting core. This is important for agent memory because a silent failure can be more damaging than an explicit refusal: if the memory state is inconsistent, the agent should expose the problem rather than hide it inside a fluent answer. Appendix D spells out the repair algorithm and retained certificate artifacts.

4 Evaluation

4.1 Evaluation Design

The evaluation separates MemGov's governance behavior from the substrates that feed it. Controlled governance cases provide known memory objects, eligibility states, and typed relationships. They isolate whether the middleware filters ineligible memories, respects must-link and cannot-link constraints, repairs satisfiable cases, refuses obstructed cases, and emits certificates. MemoryAgentBench (Hu et al., 2026) provides long contexts, questions, and answers but not MemGov objects or a formal constraint graph, so the end-to-end pipeline must retrieve evidence, extract atoms, detect relationships, and govern the resulting objects. MemGov is therefore evaluated as a governance layer after retrieval: it should preserve answer evidence when possible while reducing constraint violations and unsafe memory combinations.

Baselines. We compare the governed output against relevance-only and metadata-only baselines. We instantiate lexical retrieval with BM25, dense retrieval with Qwen3 Embedding 4B vectors, and hybrid retrieval with a BM25–embedding combination. BM25 is used as the standard lexical ranking substrate for sparse term matching (Robertson and Zaragoza, 2009). Reranked context construction uses a Qwen3 Reranker 4B model to compress high-recall candidates before extraction. Metadata filtering removes individually ineligible memories but does not repair pairwise conflicts. MemGov applies the full governance layer: eligibility constraints, must-link/cannot-link constraints, repair, refusal, and certificates.

Experimental setup and hyperparameters. All end-to-end MemoryAgentBench runs use the Conflict_Resolution split with 800 cases per atomizer setting. We use LightRAG as our base for retrieval (Guo et al., 2025). Evidence construction uses graph-expanded reranking: the system first builds a high-recall pool of 200 candidates, then reranks with Qwen3 Reranker 4B and keeps 25 chunks for query-focused extraction. Dense retrieval uses Qwen3 Embedding 4B with embedding batch size 16, and the reranker truncates each document to 500 characters. We use no query expansions or structured fact-template fallback. The agentic atomizer processes at most 25 chunks per case with a 512-token budget. Transformer models are loaded with 4-bit quantization for CUDA runs. Repair uses a fixed five-group cluster budget and local verification over every hard constraint.

Model stack. The CUDA experiments use Qwen3 Embedding 4B BNB 4-bit for dense retrieval (lainlives, 2026; Qwen Team, 2025b), Qwen3 Reranker 4B loaded with 4-bit quantization for reranking (Qwen Team, 2025c), and ModernCE base NLI for pairwise relationship detection (Miller, 2025). The end-to-end runs report four atomizer/extraction settings: Qwen3-4B-Instruct-2507 (Qwen Team, 2025a), Qwen3.6-27B (Qwen Team, 2026), Gemma-4-31B-it (Google, 2026a), and Gemma-4-E2B-it (Google, 2026b). These models instantiate the retrieval, reranking, extraction, and detection substrates around the same MemGov governance layer.

Metrics. The primary utility metric is answer hit rate: whether the returned memory set contains answer-bearing evidence. The primary safety metric is unsafe merge rate: the fraction of cases in which a method returns both sides of at least one cannot-link relationship. This is the clearest test of memory governance because two memories can each be relevant while being unsafe to use together. We also report repair success, local verification, certificate coverage, extraction success, and generated must-link/cannot-link counts. In the controlled profiles, answer hit requires retrieval of a designated expected memory ID, while the LoCoMo and LongMemEval diagnostics measure answer-alias coverage. Repair success measures the verified fraction of constrained cases. Certificate coverage includes refusals, and extraction success requires usable grounded atoms. Unsafe merge uses all evaluated cases, eligibility violations count ineligible returned objects, and M/C denotes must-link/cannot-link counts. Appendix E describes the retained audit artifacts.

Controlled eligibility construction. Across 202 cases from MemoryAgentBench's Accurate Re-

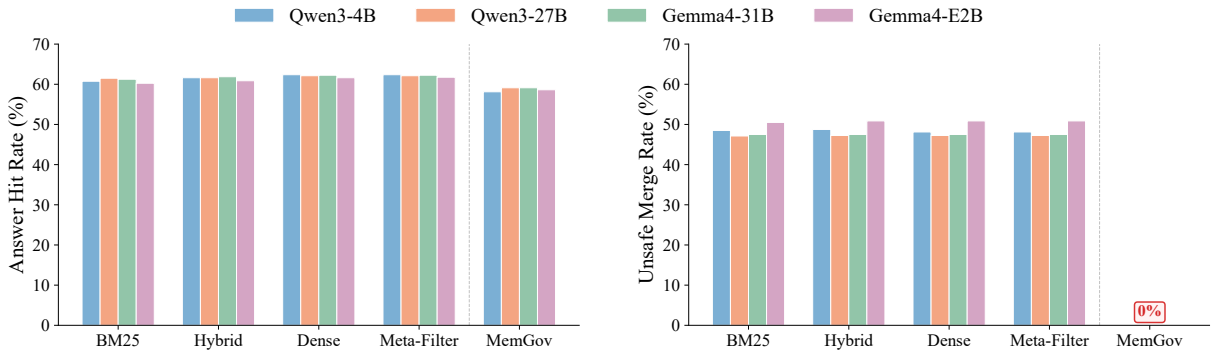


Figure 3: Utility and safety across four atomizer models, with 800 cases per model. Left: answer hit rate. Right: violations of detected cannot-links. Raw baselines reach 47–51% unsafe merges while MemGov returns none.

trieval split, we preserve the original memories and add five answer-bearing copies per case, one for each restriction in Table 2. Each copy repeats the query twice and uses the query embedding to promote retrieval. Copies inherit source timestamps, with stale and deleted labels assigned through lifecycle metadata. The 1,010 additions represent 0.219% of 460,454 case-specific memory instances and form a balanced eligibility stress test (Appendix E.2).

BM25 answer-alias diagnostics cover all 1,986 LoCoMo questions (Maharana et al., 2024) and the first 50 LongMemEval-S cases with full histories (Wu et al., 2025) (Appendix E.1).

4.2 Results and Analysis

Utility and safety. The main result is that raw retrieval and governed retrieval differ most sharply on safety, not on whether they can find relevant evidence. Figure 3 reports the multi-model MemoryAgentBench runs. Across four atomizers and 800 cases per model, MemGov keeps answer hit rate stable at 58.1–59.1%, while driving unsafe merge cases to zero. The raw lexical, dense, and hybrid baselines remain useful, but unsafe: depending on the atomizer, they violate cannot-link constraints in 47.1–50.9% of cases, corresponding to 1,924–2,395 violated pairs.

The left panel of Figure 3 shows that MemGov preserves most of the answer utility provided by retrieval. The right panel shows the governance benefit: MemGov eliminates detected unsafe combinations before the memories reach the agent. This distinction is the point of evaluating utility and safety separately. If the evaluation only measured answer hit rate, a system could appear strong while still passing conflicting memories to the agent. Unsafe merge rate captures the missing condition: whether the returned memory set is safe to use as a set, not just whether it contains a useful fact.

Unsafe retrieval case. Aggregate safety metrics correspond to a simple failure mode. For a query about the user's current home address, raw retrieval can return the current address with stale or deleted addresses because all are topically relevant. The answer is present, but the set mixes incompatible memory states. The governed pipeline filters ineligible memories, checks remaining relationships, and returns a certified safe set. Appendix Figure 6 shows the corresponding audit view. This case is representative because the unsafe memories are not irrelevant noise. They are relevant to the query, which is exactly why a ranker can select them, but relevance alone does not determine whether they may be used together.

Governance audit. The governance audit reports repair success, local verification, certificate coverage, extraction success, and the generated constraint graph. These rates are meaningful only if the benchmark produces enough constraints to exercise the repair layer. Across the four atomizer models, repair succeeds in 97.4–100% of constrained cases, certificate coverage is 100%, and local verification is complete. Extraction success ranges from 92.6% to 100%, yet the governed layer still eliminates unsafe merges. Table 1 shows the constraint workload alongside the compact audit metrics.

Across the four 800-case model settings, the detector produces 14,155 pairwise constraints: 4,196 must-links and 9,959 cannot-links. These are not incidental edge cases. Roughly three quarters of cases contain at least one enforceable constraint, and relevance-only retrieval violates cannot-links in nearly half of cases. The remaining differences

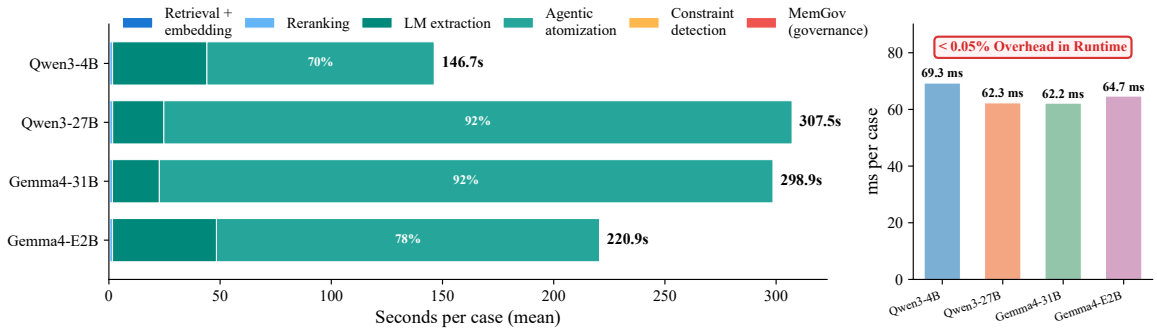


Figure 4: Runtime decomposition across four atomizer models. Workload ran on one H200 GPU for ≈ 90 hrs total. Agentic atomization dominates end-to-end cost, while the MemGov governance layer adds negligible cost.

Atomizer	MemGov hit (%)	Repair (%)	Cert. (%)	Extract. (%)	Constraints M/C	Constrained cases	Raw unsafe rate (%)	Raw violated pairs	MemGov unsafe cases
Qwen3-4B	58.1	98.4	100.0	97.6	2,097/2,522	591 (74%)	48.1–48.8	1,962–1,999	0
Qwen3-27B	59.1	100.0	100.0	100.0	681/2,106	600 (75%)	47.1–47.2	1,924–1,944	0
Gemma4-31B	59.1	98.4	100.0	100.0	577/2,439	577 (72%)	47.5	2,212–2,256	0
Gemma4-E2B	58.6	97.4	100.0	92.6	841/2,892	575 (72%)	50.5–50.9	2,298–2,395	0

Table 1: Governance audit and constraint workload over 800 MemoryAgentBench cases per atomizer. M/C denotes must-link/cannot-link constraints. Raw unsafe and raw violated pairs report the range across the relevance-only baselines: lexical, dense, and hybrid retrieval.

Stressor	Assignment	Count/share
Cross-user	Another owner	202 / 20%
Stale	Stale lifecycle	202 / 20%
Deleted	Deleted lifecycle	202 / 20%
Purpose mismatch	Incompatible use	202 / 20%
Restricted sensitivity	Restricted level	202 / 20%

Table 2: Eligibility assignments in 202 cases. Each adds one object per type (1,010 total). Shares use injected objects.

Metric	Raw	Filter	MemGov
Forbidden objects	99.9	0	0
Answer hit	0	26.2	70.3
Stale/deleted exposure	20/20	0/0	0/0
Unsafe: satisfiable	67.8	22.3	0
Unsafe: obstruction	67.8	22.3	0
Repair / cert.	—	—	100/100
Obstruction / cert.	—	—	100/100

Table 3: Controlled governance (%, 202 cases each, top-$k = 5$). Raw and MemGov use BM25. Filter uses metadata/vector retrieval. Exposure uses returned objects. Repair and obstruction use profile cases.

come from upstream object creation: atomizers differ in output-format reliability, number of grounded atoms, and the candidate pairs they expose to the detector. The table therefore shows that MemGov is not succeeding on an empty constraint graph. It is repeatedly asked to govern nontrivial working sets, and the raw retrievers repeatedly violate those constraints. The zero unsafe cases for MemGov mean that, once the relationship detector produces a hard constraint, the repair layer enforces it consistently across model settings. This is the main evidence that the reported safety gains come from constraint enforcement rather than from a weaker retrieval workload. The raw systems see the same candidate memories, but they lack the final admissibility check over the returned set.

The takeaway is that relevance-only memory can look useful while still returning unsafe combinations. Middleware governance removes those combinations without discarding most answer-bearing evidence.

Runtime. Runtime is dominated by upstream model calls. Figure 4 shows that agentic atomization accounts for 70–92% of runtime, while MemGov adds only 62–69 ms per case, less than 0.05%. Qwen3-4B is about 2× faster than the larger atomizers while achieving comparable governance quality, so future speedups should target extraction, atomization, batching, and sharding. This separation is useful experimentally because it shows that stronger retrieval or atomization can be swapped in without making the repair step the dominant cost. The governance layer remains a small boundary

Method	Answer hit	Unsafe	Cert.
BM25	60.2–61.5	47.1–50.5	—
Dense	61.6–62.4	47.2–50.9	—
Hybrid	60.9–61.9	47.2–50.9	—
Metadata filter	61.8–62.4	47.2–50.9	—
MemGov	58.1–59.1	0.0	100.0

Table 4: MemoryAgentBench baseline ranges (%) across four atomizers, 800 cases each. Table 1 gives the per-model audit.

check after the expensive evidence-construction stages. Appendices E and F provide the worked unsafe-retrieval case and implementation details.

4.3 Additional Governance Evidence

Eligibility and consistency. On eligibility stressors, metadata filtering and MemGov both block forbidden objects, but MemGov retains 70.3% answer hit versus 26.2% (Table 3). In the separate pairwise profiles, filtering leaves 22.3% unsafe merges. MemGov returns verified repairs or obstructions, with certificates for all 202 cases in each profile.

Coverage and working-set size. LoCoMo all-text answer coverage is 77.6% (Table 5). With the 50-question LongMemEval-S corpus fixed, increasing top-k from 5 to 25 raises evidence hit from 40% to 56% and answer coverage from 24% to 38%. At top-100, these rise to 66% and 48%, with zero detected unsafe merges and 100% certificates at every tested k (Table 6A).

Paired answer retention. Zero unsafe merges measures consistency, so paired answer retention is checked separately with eligibility, ranking, and corpus fixed. In the timestamp-aware top-25 run, conflict filtering reduces unsafe merges from 60% to 0%, with coverage changing from 36% to 38%: all 18 covered questions are retained and one is gained (Table 6C). The remaining coverage categories distinguish missing answer atoms from retrieval misses (Table 6B).

Temporal and multi-memory constraints. Timestamp ordering supports 25,383 temporal cannot-links, with zero detected unsafe merges (Table 7). Historical questions can require older facts, so temporal admissibility depends on the task. Conflicting values for the same subject and relation yield cannot-links, while must-link closure retains required supporting evidence. A cannot-link inside a must-link component identifies an immediate obstruction, giving a checkable reason for refusal.

Learned detector ablation. ModernCE NLI supplies 202 must-links and 202 cannot-links in the satisfiable profile. The obstruction profile has 202 must-links and 404 cannot-links (Table 8). MemGov reduces raw unsafe merges from 65.8% to zero with 100% certificate coverage in both profiles. The detector can change while repair and certification remain deterministic. Thresholds and ambiguity telemetry are in Appendix E.2.

5 Discussion and Conclusion

Discussion. Given memory objects, eligibility metadata, and typed constraints, MemGov provides a deterministic governance decision: it filters ineligible memories, repairs satisfiable constraint sets, refuses impossible cases, and produces certificates. The agent no longer receives whatever text the retriever ranked highest. It receives a memory set that passed eligibility and consistency checks, or it receives a certificate-backed refusal. This boundary matters because it separates evidence discovery from admissibility. A retriever can rank memories by relevance, but an agent memory system also needs to decide whether individually relevant memories remain safe when combined. MemGov makes that second decision explicit and auditable rather than leaving it implicit in the prompt. This also makes failures easier to diagnose, since the certificate identifies the candidate memories, constraints, and repair decision separately.

Conclusion. Agent memory therefore should not be treated only as a ranking problem. A useful memory can still be unsafe if it is unauthorized, stale, deleted, or inconsistent with another retrieved memory. This paper frames long-term language-model agent memory as governed object selection and presents MemGov as a middleware layer for enforcing explicit memory constraints. The evaluation separates answer recovery from safety and shows that relevance-only retrieval can recover answer-bearing evidence while still returning unsafe memory combinations, while MemGov removes those combinations with certificates and minimal repair overhead. Future memory systems should expose auditable constraint and repair behavior rather than placing unconstrained retrieved memories directly into model context.

Limitations

MemGov governs the memories that reach its bounded working set, but it does not guarantee that upstream retrieval finds every relevant memory or that extraction creates every correct atom. The guarantee is therefore local to the candidate memories, metadata, eligibility constraints, and pairwise constraints supplied to the repair layer. A missed contradiction can leave an unsafe pair outside the constraint graph, while a false cannot-link can separate memories that could have been safely used together. Better retrieval, extraction, and detector calibration are complementary to the governance layer rather than replaced by it.

The evaluation focuses on MemoryAgentBench conflict-resolution cases and on auditable synthetic eligibility states over extracted memory objects. This is the right setting for isolating unsafe merges, certificates, and pairwise repair, but it does not cover every deployment concern for persistent memory. Real systems also need robust identity management, user-facing controls, constraint authoring tools, logging, review workflows, and enforcement of deletion or erasure requests at the storage layer. MemGov can expose these requirements at the memory boundary, but it is not itself a complete privacy, compliance, or machine-unlearning system.

Finally, the strongest end-to-end runs depend on language-model extraction, atomization, reranking, and NLI-style detection. These stages dominate runtime and can vary across model families, quantization choices, and hardware. The repair step is inexpensive and deterministic once objects and constraints are available, but future work should improve the efficiency and reliability of object construction, expand evaluation to additional long-term-memory benchmarks, and study human-facing certificate interfaces.

Acknowledgments

OpenAI ChatGPT and other AI-assisted tools were used throughout the preparation of this manuscript to support brainstorming, organization, drafting, editing, review, and development of the experimental code. All technical claims, experimental results, equations, citations, and final manuscript content were reviewed and verified by the authors, who take full responsibility for the work.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pages 9112–9141.

Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Sugato Basu, Ian Davidson, and Kiri Wagstaff, editors. 2008. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. CRC Press.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Xiwei Chen. 2026. Epistemic memory failures in long-form narrative agents: A deployment study. ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems, OpenReview.

Pedro Chumpitaz-Flores, My Duong, Ying Mao, and Kaixun Hua. 2026. Pass: Certified subset repair for classical and quantum pairwise constrained clustering. *arXiv preprint arXiv:2601.20157*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Jonathan Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Emmimal. n.d. temporal-rag: A post-retrieval temporal layer for RAG systems. Software repository. Accessed September 30, 2026.

European Parliament and Council of the European Union. 2016. Art. 17 gdpr right to erasure ("right to be forgotten"). https://gdpr-info.eu/art-17-gdpr/.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

Google. 2026a. google/gemma-4-31b-it. https://huggingface.co/google/gemma-4-31B-it. Hugging Face model repository.

Google. 2026b. google/gemma-4-e2b-it. https://huggingface.co/google/gemma-4-e2b-it. Hugging Face model repository.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761, Suzhou, China. Association for Computational Linguistics.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 32779–32798.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2026. Evaluating memory in LLM agents via incremental multi-turn interactions. In *The Fourteenth International Conference on Learning Representations*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7969–7992.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25972–25981.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

lainlives. 2026. lainlives/qwen3-embedding-4b-bnb-4bit. https://huggingface.co/lainlives/Qwen3-Embedding-4B-bnb-4bit. Hugging Face quantized model repository.

Chingkwun Lam, Jiaxin Li, Lingfei Zhang, and Kuo Zhao. 2026. Governing evolving memory in LLM agents: Risks, mechanisms, and the stability and safety governed memory (SSGM) framework. *arXiv preprint arXiv:2603.11768*.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, Bangkok, Thailand. Association for Computational Linguistics.

David Lee Miller. 2025. dleemiller/modernce-base-nli. https://huggingface.co/dleemiller/ModernCE-base-nli. Hugging Face model repository.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Helen Nissenbaum. 2004. Privacy as contextual integrity. *Washington Law Review*, 79(1):119–158.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th*

Annual ACM Symposium on User Interface Software and Technology.

Qwen Team. 2025a. Qwen/qwen3-4b-instruct-2507. `https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507`. Hugging Face model repository.

Qwen Team. 2025b. Qwen/qwen3-embedding-4b. `https://huggingface.co/Qwen/Qwen3-Embedding-4B`. Hugging Face model repository.

Qwen Team. 2025c. Qwen/qwen3-reranker-4b. `https://huggingface.co/Qwen/Qwen3-Reranker-4B`. Hugging Face model repository.

Qwen Team. 2026. Qwen/qwen3.6-27b. `https://huggingface.co/Qwen/Qwen3.6-27B`. Hugging Face model repository.

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

Alireza Rezazadeh, Zichao Li, Ange Lou, Yuying Zhao, Wei Wei, and Yujia Bao. 2025. Collaborative memory: Multi-user memory sharing in LLM agents with dynamic access control. *arXiv preprint arXiv:2505.18279*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Alaa Saleh, Sasu Arimo Olavi Tarkoma, Anders Lindgren, Praveen Kumar Donta, Schahram Dustdar, Susanna Pirttikangas, and Lauri Lovén. 2025. Memindex: Agentic event-based distributed memory management for multi-agent systems. *ACM Transactions on Autonomous and Adaptive Systems*.

Samarth Sarin, Lovepreet Singh, Bhaskarjit Sarmah, and Dhagash Mehta. 2025. Memoria: A scalable agentic memory framework for personalized conversational ai. In *2025 5th International Conference on AI-ML-Systems (AIMLSystems)*, pages 32–39. IEEE.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. 2001. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

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

Heng Xu, Tianqing Zhu, Lefeng Zhang, Wanlei Zhou, and Philip S. Yu. 2023. Machine unlearning: A survey. *ACM Computing Surveys*, 56(1).

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, volume 38, pages 17577–17604. Curran Associates, Inc.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. 2026. Lifelong learning of large language model based agents: A roadmap. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: enhancing large language models with long-term memory. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

A Motivating Examples and Object Flow

This appendix gives supporting material for the main paper. It is organized in the same order as the main text: motivating examples, formal details, construction and detection, repair and certificates, additional evaluation artifacts, and implementation/model details. The examples below expand the same boundary used throughout the paper: retrieval proposes candidate evidence, while governance decides which memory objects may be used together. This material supplements Sections 3.1 and 3.2.

A.1 Running Memory Example

Consider an agent that stores several memories about a user's travel preference. One memory states that the user prefers direct flights. A newer memory states that the user now prefers the cheapest option. A deleted memory contains an old airport preference. A raw retriever may return all three because each is relevant to travel planning. The governed formulation treats these as memory objects. The deleted memory is ineligible. The two active preferences may be linked if they are compatible or separated if they express conflicting instructions for the same decision. The final answer should be based only on a memory set that satisfies eligibility and consistency.

This example separates two concerns: retrieval determines which memories should be considered, while governance determines which of those memories may be used together.

A.2 Multi-Hop Object Flow

Consider a question that requires identifying an author, the author's spouse, and the spouse's country of citizenship. A retrieval system may select chunks stating that a book was written by a particular person, that this person is married to another person, and that the spouse is a citizen of a country. The extractor converts these chunks into memory atoms. If the selected context also contains a contradictory author or citizenship statement, the detector can propose a cannot-link constraint. The repair layer then ensures that the returned memory set does not contain incompatible memories together.

A.3 Address Repair Example

Suppose the retrieved memories include a current address, an older address, and a deleted address. Raw retrieval may return all three if they are close to the query. The governed pipeline first excludes the deleted memory. It then evaluates whether the current and older memories can safely be returned together. If the older memory is stale or contradicts the current memory, the repair layer separates the pair or returns only the safe subset. The certificate records which memories were considered, which constraints were checked, and whether the final memory set passed verification.

B Formalization of Governed Memory Selection

This section expands the object and atom notation and constrained selection objective in Section 3.1.

B.1 Memory Objects and Atoms

Let a memory store contain objects $\mathcal{M} = \{m_1, \ldots, m_n\}$. Each memory object has textual content $\tau(m_i)$, a representation vector $v(m_i)$, and metadata describing ownership, purpose, state, sensitivity, trust, source, and time. Each object may contain one or more extracted atoms. An atom records a compact factual claim: subject, relation, object value, evidence span, confidence, provenance, and governance metadata. For example, a chunk stating that a person is a citizen of a country may produce an atom whose subject is the person, whose relation is citizenship, and whose object value is the country.

Atoms make memory inspectable and make constraint generation possible. Two memories can be compared by their normalized subject and relation. If they describe the same subject and relation but disagree on the value, they become candidates for cannot-link detection. If they describe the same fact, they become candidates for must-link detection.

B.2 Constrained Objective

The eligible set for a governance context γ is

$$\mathcal{M}_\gamma = \{m_i \in \mathcal{M} : E(m_i, \gamma) = 1\}.$$

The governed selection problem can be written as:

$$\max_{R \subseteq \mathcal{M}} \sum_{m_i \in R} s(q, m_i)$$

subject to

$$R \subseteq \mathcal{M}_\gamma,$$
$$\{m_i, m_j\} \nsubseteq R \quad \forall (m_i, m_j) \in \mathcal{D}.$$

Must-link relationships add repair constraints over the assignment used internally by the repair layer:

$$(m_i, m_j) \in \mathcal{L} \quad \Rightarrow a(m_i) = a(m_j),$$
$$(m_i, m_j) \in \mathcal{D} \quad \Rightarrow a(m_i) \neq a(m_j).$$

Here $s(q, m_i)$ may be supplied by any retrieval substrate. The governed layer determines whether the proposed high-scoring memories are admissible.

C Memory Construction and Constraint Detection

This section supplements Section 3.3 with diagnostic extraction settings and telemetry that distinguish object creation failures from repair failures.

C.1 Template Fallback as a Diagnostic

Earlier diagnostic runs used deterministic fact templates to recover structured facts from benchmark sentences. That was useful for isolating retrieval and repair behavior, but it does not represent how a general agent would create memory. The realistic pipeline therefore avoids structured fallback and uses language-model extraction and atomization instead. Template-based extraction remains useful as a diagnostic because it can test whether constrained repair works when the correct objects are already available.

C.2 Detection and Extraction Telemetry

The evaluation records extraction telemetry and detector telemetry alongside aggregate metrics. Extraction telemetry includes atom counts, answer-bearing atom counts, output-format repair counts, and issue counts. Detector telemetry includes candidate pair counts, classification counts, ambiguous pair counts, and generated must-link/cannot-link counts. These details are necessary for interpreting aggregate safety metrics because the governance layer can only enforce the constraints it receives.

D Repair Algorithm and Certificates

This section details the constraint compilation and certificate verification steps summarized in Section 3.4.

D.1 Repair Algorithm Details

The repair implementation compiles constraints over the active working set. Must-link constraints are closed with union-find, so each must-link component is treated as a unit. The compiler then checks cannot-link constraints. If a cannot-link falls inside a must-link component, or if a hard constraint cannot be evaluated over the working set, the run is marked obstructed.

If compilation succeeds, the repair layer calls a seeded pairwise-constrained assignment routine on the memory embeddings, using the compiled must-link and cannot-link pairs as hard constraints. Experiments use a fixed random seed, a fixed cluster budget, and deterministic solver settings. The resulting assignment is then verified by checking every hard cannot-link in the working set against the returned labels. The certificate records the working set, constraint counts, assignment, residual violations, and local verification result.

Figure 5 connects these steps. Panel A follows evidence into an atom and memory object. Panel B shows a feasible assignment and its verified repair certificate. Panel C shows an incompatible pair inside a must-link component and the corresponding obstruction certificate.

D.2 Retained Audit Artifacts

Each run writes aggregate metrics and runtime telemetry alongside per-case records of the question, answer aliases, selected and expected evidence chunks, extracted atoms, memory objects, generated constraints, baseline retrievals, violation flags, and repair certificates. These records support both aggregate tables and the qualitative examples in the main text. For a reported violation, they identify the retrieved memories, violated constraint, governed memory set, and certificate.

E Additional Evaluation Artifacts

This section supports Section 4 and Section 4.2. It records the retained audit view used to explain the unsafe-retrieval example in the main text. Figure 6 shows a worked unsafe-retrieval case with initial retrieval and repaired memory set.

E.1 Broader Coverage and Constraint Diagnostics

Cross-benchmark coverage. Table 5 compares LoCoMo conversation-only and all-text atomization with the LongMemEval-S diagnostic. Denser LoCoMo atomization improves answer coverage while lowering exact evidence-session recall. These metrics measure evidence availability separately from generated answer accuracy.

Working-set sensitivity and coverage losses. Table 6A reports every tested working-set size. In-

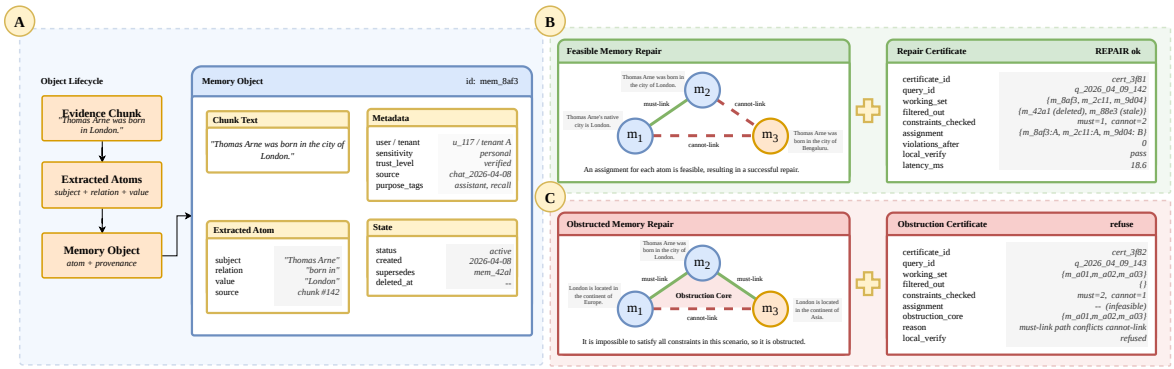


Figure 5: Object construction (A), feasible repair with a verified assignment (B), and an obstructed constraint graph with a refusal certificate (C). Must-links join compatible memories, while cannot-links separate incompatible memories.

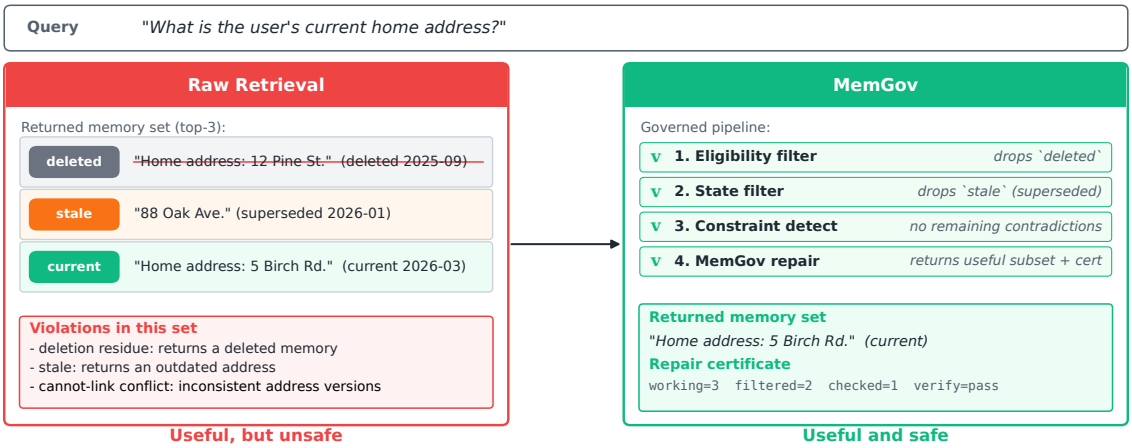


Figure 6: Worked unsafe-retrieval case. Raw retrieval finds answer-bearing evidence but also returns stale or deleted memories. The governed pipeline applies eligibility and consistency checks before returning memory to the agent.

creasing LongMemEval-S top-k to 100 raises evidence hit to 66% and answer coverage to 48%, with zero detected unsafe merges and complete certificate coverage across all tested settings. Table 6B reports the exclusive top-25 coverage categories. At top-25, 24 questions lack a matching answer atom in their annotated evidence sessions, four miss those sessions, and three retrieve a session without its matching answer atom.

Paired answer retention. Table 6C holds the corpus, eligibility, ranker, and top-25 budget fixed. Cannot-link filtering reduces unsafe merges from 60% to zero while retaining all 18 previously covered questions, losing none, gaining one, and leaving 31 uncovered. This comparison directly tests evidence retention alongside consistency enforcement. Reproduction settings are in Appendix E.2.

Timestamp-supported constraints. Table 7 distinguishes general slot conflicts from the timestamp-supported subset. Historical questions can require older facts. Timestamp-supported cannot-links prevent incompatible versions from being used together without imposing a universal newest-value rule.

Learned constraint detection. Table 8 tests the learned detector interface. ModernCE NLI uses contradiction threshold 0.85 and entailment threshold 0.75. Raw BM25 has 65.8% unsafe merges, while MemGov has zero and complete certificate coverage in both controlled profiles. The detector supplies typed relations, and the same repair layer checks their joint satisfaction. In the NLI obstruction profile, one additional cannot-link per case is injected to conflict with a detected must-link.

Multi-memory support and obstruction. Section 4.3 explains how pairwise decomposition and must-link closure represent multi-memory conflicts and supporting evidence. The compilation steps

are detailed in Appendix D.1.

E.2 Diagnostic Settings and Reproduction

The controlled Accurate_Retrieval profiles use 202 cases with a locatable answer and at least two candidate chunks, BM25 ranking for MemGov, 256-dimensional hashing embeddings, and top-$k = 5$. The eligibility profile appends five stressors to every case while preserving its original candidates. Each type accounts for 202 of the 1,010 appended objects. Across all case-specific candidate sets, 459,444 original objects plus these additions give 460,454 memory instances. The injection fraction is therefore $1{,}010/460{,}454 = 0.219\%$. This denominator counts repeated objects in different question-specific sets, rather than unique corpus objects. Lifecycle labels are interventions on copied objects, and time is inherited from the source object used by the builder. No elapsed-time cutoff assigns stale or deleted labels. Table 2 reports the balanced assignment, while Table 3 reports retrieved-object exposure. Each stressor appends a descriptive tag and source text after two copies of the query, and its vector is the query embedding. Stale and deleted exposure rates use returned objects as the denominator. Thus, 20% retrieved exposure for each type is distinct from the 0.219% overall injection fraction.

Worked eligibility and refusal cases. In the first saved controlled eligibility case, BM25 returns exactly the five stressors: deleted, restricted-sensitivity, stale, cross-user, and purpose-mismatched memories. MemGov excludes all five and returns original eligible memories. Their query relevance therefore does not determine their admissibility. In the first obstruction case, the certificate records an empty assignment and the reason "cannot-link constraint falls inside a must-link component." The record identifies the incompatible pair, making the refusal checkable from the supplied constraints.

The satisfiable profile adds compatible supporting evidence and incompatible alternatives. The obstruction profile deliberately places a cannot-link inside a must-link component. These profiles isolate enforcement with known constraints.

Benchmark and atomization settings. LoCoMo diagnostics use the complete 1,986-question QA collection. The all-text export includes 3,074 source records and produces 3,242 atomized chunks. LongMemEval-S uses the first 50 questions and every session in their histories, with 2,375 distinct source sessions. Both newer atomizations use the fixed construction and evaluation settings in Table 9. Local evaluation requires no model generation. Answer coverage matches normalized aliases against atom text, object values, and evidence spans. Evidence-session recall uses the benchmark's session annotations. The 50 questions are from the single-session-user category. Their 11,109 chunks contain 60,609 memory objects.

Paired comparison reproduction. The paired LongMemEval-S diagnostic fixes the corpus, eligibility predicates, top-25 ranker, and timestamp-aware constraints, then toggles cannot-link filtering. Its governed memory IDs match the saved slot-plus-temporal run for every question. The recorded comparison distinguishes retained, lost, gained, and uncovered answers and records input hashes. The reproducibility script `scripts/analyze_memgov_paired_utility.py` accepts the saved report and atomized JSONL as inputs. The resulting paired report supports Table 6.

Detector thresholds and ambiguity telemetry. The NLI variant uses ModernCE base NLI with contradiction threshold 0.85. At entailment threshold 0.90, it leaves 202 same-value pairs ambiguous. Threshold 0.75 recovers those must-links and leaves zero ambiguous pairs in both reported profiles. This controlled calibration evaluates the detector interface and certificate behavior (Table 8).

F Implementation and Model Details

This section explains the replaceable retrieval, extraction, detection, and execution components in Sections 3.2, 4, and 4.2. Concrete model identifiers are listed in Section 4.

F.1 Retrieval Substrate

The implementation supports several retrieval modes. Lexical retrieval is used as a transparent factual baseline. Dense retrieval uses learned representations to find semantically related memories. Hybrid retrieval combines both signals. Graph-expanded retrieval follows entity and relation chains in the memory context, which is especially useful when a question requires multiple evidence steps. Reranking is used to compress a high-recall candidate set into a smaller context for extraction.

Diagnostic	QA cases	Chunks	Top-k	Evidence hit	Session recall	Answer coverage
LoCoMo conversation-only	1,986	440	5	89.6	84.6	46.8
LoCoMo all-text	1,986	3,242	5	53.9	50.7	77.6
LongMemEval-S	50	11,109	25	56.0	56.0	38.0

Table 5: Cross-benchmark coverage diagnostics (%). The MemoryAgentBench comparison is in Table 4. LongMemEval-S has 60,609 memories and 46% matching-answer availability in annotated evidence sessions. Chunks are atomizer inputs, rather than QA cases or source sessions.

A. LongMemEval-S top-k sensitivity

k	Hit	Recall	Answer	Cert.	Unsafe
5	40	40	24	100	0
10	48	48	32	100	0
25	56	56	38	100	0
50	58	58	42	100	0
100	66	66	48	100	0

B. Top-25 answer-coverage categories

Category	Cases	Rate
Answer covered	19	38
No matching answer atom	24	48
Expected session missed	4	8
Session found, answer atom missed	3	6

C. Paired top-25 comparison on the timestamp-aware run

Filter	Answer coverage	Unsafe	Retained	Lost	Gained	Uncovered
Before cannot-link filtering	36	60	—	—	—	—
After cannot-link filtering	38	0	18	0	1	31

Table 6: LongMemEval-S diagnostics over 50 QA cases. Rates are percentages and transitions are case counts. Panel A uses 36,838 detected cannot-links. Panel B categorizes returned evidence using answer-alias matching. Panel C verifies the governed IDs against the saved timestamp-aware run and directly measures retained, lost, and gained coverage.

Mode	CL	Temporal	Answer	Unsafe
Slot	36,838	—	38	0
Temporal only	25,383	25,383	36	0
Slot + temporal	36,838	25,383	38	0

Table 7: Timestamp-supported constraints on LongMemEval-S at top-25. All 60,609 memory objects have parsed dates. CL counts cannot-links, and Temporal counts the timestamp-supported subset. Answer coverage and unsafe merge are percentages.

Profile	ML/CL	Raw	MemGov	Cert.
Satisfiable	202/202	65.8	0	100
Obstruction	202/404	65.8	0	100

Table 8: NLI-controlled profiles over 202 MemoryAgentBench cases. ML/CL counts must-links/cannot-links. Raw BM25 and MemGov columns give unsafe-merge rates, and Cert. gives certificate coverage (%). Both profiles have zero ambiguous pairs. Telemetry and thresholds are in Appendix E.2.

These retrieval choices are treated as substrates rather than as the main contribution. A stronger retrieval substrate can improve evidence coverage, but it still needs governance before returned memories are used by an agent.

F.2 Extraction and Atomization Models

The extraction stage uses instruction-tuned language models to produce memory atoms from selected evidence. Larger models are useful for query-focused extraction, while smaller models can serve as faster agentic atomizers over individual chunks. These models are practical implementation choices rather than the source of the governance guarantee.

The atomization stage is one of the main runtime bottlenecks. It can require many model calls when each selected chunk is processed separately. This cost is acceptable for strong experiments and qualitative analysis, but larger runs benefit from parallel execution, smaller atomizer models, and high-memory GPU nodes.

F.3 Constraint Detection Models

The detector stage uses natural language inference models or language-model classifiers to label relationships between candidate memory pairs. The detector thresholds are fixed before evaluation rather than adapted per case. This makes the evaluation easier to interpret because the detector operates at one consistent setting throughout the benchmark.

Stage	Fixed setting
Atomizer	Qwen3.6-35B
Chunk input limit	2,500 characters
Generation budget	2,048 tokens
Local ranker	BM25
Local embeddings	256-dimensional hashing

Table 9: Shared settings for the LoCoMo all-text and LongMemEval-S atomization diagnostics. Benchmark scope and metric matching rules are given in Appendix E.2.

F.4 Local and High-Performance Execution

The project supports both local laptop execution and high-performance GPU execution. Local runs are useful for smoke tests, debugging, and qualitative examples. Larger GPU runs are needed for stronger extractors, stronger atomizers, and full benchmark shards. Quantized models are used to fit large extraction and reranking models into available GPU memory while keeping the pipeline practical for repeated experiments.

F.5 Saved Artifacts

Appendix D.2 lists the aggregate reports, runtime telemetry, per-case audits, and certificates saved by each run.