

When Should LLM Agents Forget? Memory Filtering Under Query-Critical Resolver Errors

Samarth Vinayaka

University of Florida

Gainesville, FL, USA

samarthvinayaka@ufl.edu

Abstract

LLM memory systems often filter facts classified as stale from the current prompt. We study this query-time context filtering, without deleting stored memories, in memory-conditioned question answering. Across three models on a synthetic benchmark, controlled label corruption moves hard-drop from gains of +0.186 to +0.230 at perfect resolution to losses of -0.176 to -0.181 at the lowest tested reliability. Paired targeted interventions show greater degradation from false-drop than false-keep; retaining evidence can still fail when incorrect labels steer generation. On one frozen 102/102 calibration/test query split, empirical policy selection using labeled calibration outcomes reaches 0.655 accuracy versus 0.641 for always hard-drop and 0.662 for test-informed oracle selection. A question-disjoint MemoryAgent-Bench replication is inconclusive: gain +0.03, 95% CI [-0.01, +0.08]. These results establish a controlled sensitivity mechanism, not generalization to long-horizon agents. Global-accuracy crossovers require gold labels and are model- and task-specific diagnostics. Context filtering should be cost-sensitive and validated on matched labeled data; label-free retention is an accuracy fallback under the tested errors, subject to separate privacy and retention requirements.

1 Introduction

LLM agents deployed over long sessions or across many interactions accumulate memory: facts about a user or environment that were true when written but may not remain true. A user who says “I live in Boston” this month may say “I moved to Austin” next month; a project deadline expires; a stated preference gets walked back and later re-asserted. An agent that treats all remembered facts as equally current will, sooner or later, act on stale information. We show that resolution accuracy and downstream behavioral accuracy can dissociate in

both directions: a resolver can be perfect and the agent can still answer incorrectly, and an assembly strategy that helps under a reliable resolver can actively harm under an unreliable one.

Scope of “forgetting.” Throughout, *hard-drop* means query-time context filtering: excluded memories remain in the persistent store and can reappear for a later query. We evaluate memory-conditioned question answering, not permanent erasure, multi-turn trajectories, planning, or tool use. The agent-memory motivation therefore exceeds the directly tested task scope.

A substantial body of work addresses this by improving *resolution*: classifying each memory as current, superseded, or expired, via a write-time consolidation step, a retrieval-time re-ranking step, or an LLM-based conflict-resolution judgment. This line of work often implicitly assumes that better resolution accuracy translates fairly directly into better downstream behavior.

We show this assumption does not hold in general. A resolver can be perfect and an agent can still answer incorrectly, if the packaging of that resolved information does not make the correction usable by the generator. Figure 1 schematically illustrates how a generator can respond differently to the same correctly resolved facts under different assembly policies.

We formalize this by decomposing the pipeline into three independently manipulable stages – **Resolver**, **Assembly**, **Generator** – and holding the resolver and generator fixed while varying only assembly format. Prior work often couples resolution and assembly into a single system, making it difficult to attribute a behavioral improvement to better resolution versus better packaging of the same resolution; our design isolates assembly as an independently manipulated variable.

Our central empirical contributions are:

1. **A resolver-error sensitivity test.** Corrupt-

Memory chain (query: “Where does the user live?”)	
Jan 1:	“I live in Boston.”
Feb 1:	“I moved to Austin.”
Flat context (concatenated, validity ignored):	
“The user lives in Boston. The user moved to Austin.”	
Generator output: “ Boston ” – incorrect	
Structured context (validity-typed, non-current dropped):	
“ <i>CURRENT</i> : The user moved to Austin.”	
Generator output: “ Austin ” – correct	

Figure 1: Schematic illustration of the hypothesized mechanism, not a recorded model response. Both contexts use the same correctly resolved chain. Exact assembly templates are in Appendix L.

ing oracle labels at a known rate moves AssemblyGain through zero on all three models; targeted interventions show that dropping a query-critical current fact is substantially more damaging than retaining a stale one (§6).

2. **Policy alternatives under resolver error.** Soft retention distinguishes deletion loss from label-induced steering but is not automatically safe; a calibration-selected hard-drop/flat policy has low test-informed oracle regret on held-out queries within matched corruption regimes (§6).
3. **Oracle and published-benchmark diagnostics.** Oracle and format ablations isolate the benefit of filtering when labels are trustworthy; a tuned diagnostic on the published MemoryAgentBench benchmark exhibits the predicted extraction–deletion mechanism but is not presented as held-out external validation (§7–§9).

The benefit of validity-aware assembly is *mediated* by upstream resolver errors: filtering improves behavior when labels are accurate and can harm it when errors remove needed evidence. Structured, content-dropping assembly is therefore not strictly dominant over flat concatenation. The operational quantity is not global label accuracy alone, but the cost and incidence of query-critical false drops.

2 Related Work

We position this work as orthogonal to memory-system design: we do not propose a new memory store, retrieval mechanism, or consolidation algorithm. We isolate and characterize one interface – how already-resolved validity information

is packaged into a generator’s context – that existing systems bundle with their own resolution or retrieval mechanism, making it hard to attribute their reported gains specifically to packaging.

Agent memory systems. MemGPT-style paging (Packer et al., 2023) moves content between working and external context via LLM function calls, using “in context” as a proxy for relevance and recency rather than validity. Mem0 (Chhikara et al., 2025) performs write-time classification of each new fact against existing memories (add/update/delete/no-op). A-Mem (Xu et al., 2025) organizes memory via dynamic, Zettelkasten-style linking, optimizing retrieval relevance. Generative Agents (Park et al., 2023) and Reflexion (Shinn et al., 2023) both treat memory as an evolving textual or episodic record, without an explicit validity-typed resolution step. To our knowledge, none of these vary assembly format independently of their own retrieval or resolution mechanism, so a behavioral gain due to better resolution cannot be separated from one due to how resolved content is packaged.

Sanity check against an end-to-end memory system. We ran Mem0 OSS v2.0.11 end-to-end on an $n = 102$ VALIDITYMEM-530 subset; the version was recovered from the unchanged environment’s installed-package metadata and timestamped before output creation. Mem0 used Claude Haiku 4.5 internally, FastEmbed with BAAI/bge-small-en-v1.5, and local Qdrant. Under the frozen grader, Mem0 reaches 0.480 versus subset-matched oracle-flat 0.578 and oracle-hard-drop 0.745. By scenario it scores 0.941 on `historical_only`, 0.588 on `superseded_monotonic`, 0.353 on `expired_interval` and `current`, 0.412 on `implicit_conflict`, and 0.235 on `non_monotonic_reversion`. We retain it as a sanity check rather than a controlled system comparison because retrieval, resolution, and assembly remain coupled.

Context compression. LongLLMLingua (Jiang et al., 2023) performs query-conditioned, perplexity-based compression that is validity-agnostic by design – it can compress away a needed correction as readily as anything else. Our `structured_budget` condition (Appendix A) studies validity-aware budget selection specifically, and shows that even a validity-aware selector can fail by dropping the fact that matters under a tight budget.

Benchmarks. MemoryAgentBench (Hu et al., 2025) evaluates four memory competencies – accurate retrieval, test-time learning, long-range understanding, and selective forgetting – via reconstructed and newly-introduced datasets including FactConsolidation, which we reuse as a published-benchmark diagnostic (§9). It reports end-task accuracy without a resolver-versus-assembly decomposition. We call our controlled diagnostic VALIDITYMEM-530 to avoid collision with the previously published FreshBench temporal-generalization benchmark (Zhu et al., 2025).

Adjacent memory-conflict and consolidation work. MemConflict evaluates dynamic, static, and conditional memory conflicts through black-box answers and white-box retrieval/ranking analysis (Tao et al., 2026). Selective QA over conflicting multi-source personal memory explicitly separates evidence availability from conflict resolution and abstention (Yang et al., 2026). StateFuse directly contrasts conflict-preserving and collapsed memory surfaces (Volkov et al., 2026). These studies establish that conflict structure and preservation matter; our complementary design holds candidate evidence fixed, manipulates packaging independently, and causally sweeps resolver errors until the behavioral effect reverses. Concurrent work also compares a resolver with a deterministic max-serial baseline on MAB FactConsolidation (Reddy and Challaram, 2026); STALE (Chao et al., 2026) benchmarks resolution without a decomposed assembly factor; and Memanto (Abtahi et al., 2026), TiMem (Li et al., 2026), and Control-Plane Placement (Yang, 2026) study end-to-end typed memory or the placement of forgetting logic rather than an independently randomized packaging policy.

What distinguishes this work: (1) an experimental design that holds resolution quality fixed and varies assembly policy; (2) interventional evidence that a content-dropping policy’s benefit is conditional on upstream reliability and can reverse sign; and (3) separate reporting of resolution, query-critical false drops, evidence retention, and downstream behavior rather than one end-task number.

3 Problem Formulation

Let $M = \{m_1, \dots, m_k\}$ be the set of candidate memories relevant to a query q issued at time t . Each memory has a ground-truth validity state $V(m)$, unknown to the system at time t , taking one of five values: *current*, *superseded*,

expired, *historical*, or *conflicting*. We take M as given, selected by an upstream *retrieval* step outside our scope; an end-to-end system’s retrieval, resolution, and assembly stages can confound one another (§2 examines this directly), which is exactly why we hold M fixed and vary only the stages below.

A **Resolver** $R : (M, t) \rightarrow \hat{V}$ produces a predicted validity label for each memory. An **Assembly** function $A : (M, \hat{V}, q, t) \rightarrow c$ applies a query- and time-dependent retention and presentation policy, producing a context string c passed to a **Generator** $G : (c, q) \rightarrow a$. Query dependence is necessary: for example, a historical memory is retained for a historically scoped query but not treated as current evidence for a present-state query.

We report two metrics, kept explicitly separate rather than collapsed into one end-task number, plus two derived quantities used throughout:

$$\text{ResolutionAcc} = \Pr[\hat{V}(m) = V(m)]$$

computed over every candidate memory (not just whether the single current fact was identified – a resolver that labels everything *current* would score perfectly on a recall-only variant of this metric without resolving anything). ResolutionAcc requires gold labels and is therefore a retrospective experimental diagnostic, not an observable deployment gate. We also report

$$\text{BehavioralAcc} =$$

$$\Pr[G(A(M, R(M, t), q, t), q) = \text{gold}(q)]$$

the standard downstream task accuracy. Because these are measured at different levels (memory-level vs. query-level), we do not report their difference as a single scalar; we instead report:

$$\text{OracleShortfall} = 1 -$$

$$\text{BehavioralAcc}(A_{\text{flat}} \mid R_{\text{oracle}})$$

how far even a perfectly-resolved, flat-assembled system falls short of ceiling, and, our central quantity of interest,

$$\text{AssemblyGain} = \text{BehavioralAcc}(A_{\text{hard}})$$

$$- \text{BehavioralAcc}(A_{\text{flat}})$$

computed within a fixed resolver tier, isolating the effect of assembly format under controlled resolver and generator settings.

Experimental design. We cross four resolver tiers with assembly modes, holding the generator fixed, to isolate assembly as an independently manipulated variable:

Resolver tiers: `no_labels` (a floor baseline with no validity signal), `oracle` (gold labels), `deterministic` (rule-based temporal ordering), and `hybrid` (deterministic rules with LLM adjudication escalation for ambiguous cases). §6 additionally sweeps a fifth, continuous condition: oracle labels corrupted at a controlled rate p .

Assembly modes: Table 1 defines the five conditions and their implementation names. All omission is local to the current prompt; none of these policies modifies the memory store. Historical retention depends on query intent. Exact headers, item labels, ordering, and separators are given in Appendix L.

Because `flat` ignores validity by construction, its `BehavioralAcc` is identical across all resolver tiers – intentional, not an artifact: a shortfall under oracle-resolved `flat` context cannot be caused by incorrect validity labels, but may reflect both context presentation and the generator’s ability to use the evidence. This is not a deployment claim; it isolates the upper bound of assembly’s utility when validity labels are trustworthy, which we then relax in §6–§9.

4 ValidityMem-530

We construct a synthetic benchmark, `VALIDITYMEM-530`, to obtain gold validity labels for controlled analysis. We avoid the name `FreshBench` because it already denotes a NAACL 2025 temporal-generalization benchmark (Zhu et al., 2025); the released Python package retains its legacy `freshbench_lite` directory name only for code compatibility. The benchmark comprises 530 queries over 1,936 unique stored memory objects across six scenario types: `current` (no conflict), `superseded_monotonic` (a temporally monotonic update chain), `expired_interval` (a closed validity window), `historical_only` (paired present- and past-scoped questions over historical evidence), `non_monotonic_reversion` (a value is stated, changed, then reverted), and `implicit_conflict` (conflicting statements with no explicit correction language).

Example. A `superseded_monotonic` chain: *Jamie*: “*I work remote.*” then *Jamie*: “*I*

just switched to working in-office.” – gold answer *in-office*, first statement labeled `superseded`. `implicit_conflict` instances instead pair an explicit statement with a later, behaviorally inconsistent one and no correction language (e.g. a stated vegetarian later mentioning eating meat), probed with a yes/no conflict question.

Construction and statistics. Memories and queries are generated from a templated vocabulary of entities, slot values, and phrasing variants, with timestamps in varying styles. Each query references 9–11 candidates (mean 10.31): a relevant chain plus seeded, interleaved distractors. The 1,936 figure counts unique stored objects; reuse across paired queries yields 5,462 query–memory candidate occurrences. Scenario and question type are different axes: `superseded_monotonic`, for example, contributes extraction, yes/no, and aggregation queries. Their exact cross-tabulation is: current 40 extraction; superseded 60 extraction, 60 yes/no, 34 aggregation; expired 80 extraction, 40 yes/no; historical-only 40 present-state extraction and 40 historical; reversion 50 extraction and 50 yes/no; and implicit conflict 36 probes. These sum to extraction 270, yes/no 150, historical 40, probe 36, and aggregation 34. We release the generator and consistency checks; `VALIDITYMEM-530` is a controlled diagnostic, not a replacement for naturalistic evaluation.

5 Experimental Setup

Experiments use OpenAI `gpt-4o-mini`, Anthropic `claude-haiku-4-5-20251001` (Haiku 4.5), and `claude-sonnet-5` (Sonnet), the exact API identifiers recorded for the runs. Unversioned aliases do not establish an immutable model snapshot. The two smaller models run at temperature 0; Sonnet rejects fixed-temperature sampling, and we disable its default adaptive thinking to match the direct-answer setup. Generation uses a 60-token cap. The same generation and adjudication prompts are used across models. The 6k `MemoryAgentBench` extraction prompt was iteratively refined after inspecting GPT-4o-mini failures on that sample, so those results are explicitly a tuned diagnostic; the subsequent question-disjoint 32k run freezes both extraction and generation prompts (§9). Hosted APIs are not exactly deterministic across time. We report paired, query-level bootstrap confidence intervals (2,000 resamples; fixed seed), conditional on the saved

Paper name	Code mode	Context construction
Flat	flat	All memories in fixed input order; no validity labels or headers.
Hard-drop	structured	Current, then uncertain/conflicting, then historical if historically queried; superseded/expired omitted.
Soft-retain	structured_flagged	All memories; current first, then uncertain/conflicting, then explicitly labeled non-current evidence.
Format-only	structured_nofilter	All memories in input order beneath one “All Facts” header, with bullets; validity ignored.
Budgeted hard-drop	structured_budget	Hard-drop eligibility and section order with a whole-line whitespace-word budget.

Table 1: Assembly conditions in one place. Soft-retain and hard-drop are saved as `soft_retain` and `hard_drop` in the matched sweep. All non-budgeted comparisons retain every eligible item.

generations and corruption realizations. Related questions may share memory chains; these intervals neither account for that clustering nor estimate across-seed uncertainty. All stored responses were regraded after freezing an abstention-aware committed-answer parser; a new blinded audit agrees on 118/120 responses (98.3%, Wilson 95% CI [94.1, 99.5], $\kappa = 0.962$; Appendix H).

6 Resolver-Error Sensitivity Sweep

The oracle table uses all 530 queries. To control API cost, every corruption point uses a fixed $n = 204$ subset: the first 34 queries in frozen file order from each of six scenario types, generated with benchmark seed 0 and selected before model calls. File order was not randomized, so we call this a scenario-balanced sensitivity subset, not a representative sample; Appendix C reports its question-type composition. The original multi-model intervention uses corruption base seed 0; the matched three-policy rerun uses portable SHA-256-derived per-query seeds. Starting from gold labels, we independently replace each relevant-chain label with a uniformly random different state at $p \in \{0, .1, .2, .3, .4, .5\}$ and report realized ResolutionAcc averaged over relevant-chain accuracies per query; distractors remain labeled current. The intervention can also produce the resolver state `uncertain`. This deliberately broad sensitivity intervention is not a generative model of realistic resolver errors; targeted false-drop and false-keep interventions separately probe error structure. A cached `flat` answer per query removes between-condition baseline sampling variation; it does not eliminate uncertainty in the sampled baseline.

At perfect resolution, hard-drop gains are +0.186, +0.186, and +0.230 for GPT-4o-mini, Haiku, and Sonnet; at the lowest tested reliability they are −0.181, −0.176, and −0.176. As-

semblyGain decreases monotonically as reliability degrades, with all three curves crossing zero. Linear interpolation places the diagnostic crossings near ResAcc 0.758, 0.668, and 0.688, respectively, but these are model- and task-specific summaries rather than deployment thresholds. Appendix Table 11 further shows why: heterogeneous resolvers with equal or higher global accuracy do not necessarily yield the gain implied by the randomized curves. Error structure, especially query-critical false drops, is the more operational concern.

Non-deleting validity awareness is not automatically safe. We ran the complete matched three-policy sweep on GPT-4o-mini: label-free flat, soft-retain (retain all evidence with validity labels and ordering), and hard-drop. Soft-retain keeps current-fact recall at 1.0 throughout, whereas hard-drop recall falls to 0.721 as query-critical false-drop rises to 0.279. Nevertheless, soft-retain’s gain declines from +0.157 [0.093, 0.216] to −0.118 [−0.196, −0.044]. At ResAcc 0.612, both soft-retain (−0.118) and hard-drop (−0.098) significantly underperform flat. Preserving content prevents deletion loss but not misleading-label steering; under these tested corruptions, label-free flat yields higher answer accuracy when the supplied validity guidance is unreliable.

False-drop is costlier than false-keep. Targeted Haiku corruption probes error direction using one shared, regraded flat baseline. False-keep changes eligible non-current labels to current; false-drop changes eligible current labels to a random non-current state. Because uncertain/conflicting labels remain retained, the intervention probability is not itself the realized omission rate. At $p = .2$, Gain is +0.180 [0.107, 0.260] for false-keep versus +0.047 [−0.040, 0.140] for false-drop; the paired false-keep-minus-false-drop difference is +0.133 [0.073, 0.193]. At $p = .4$, the gains are +0.107 [0.033,

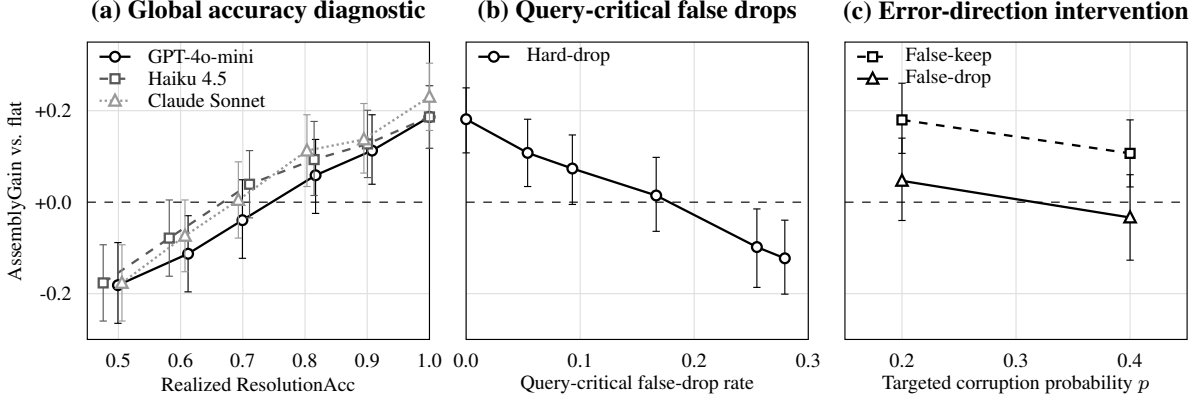


Figure 2: The mechanism is not captured by global accuracy alone. (a) Hard-drop gain versus gold-dependent ResolutionAcc ($n = 204$ per point). (b) On the matched GPT-4o-mini sweep, gain declines with query-critical false-drop rate ($n = 204$). (c) Targeted Haiku corruption shows false-drop is costlier than false-keep ($n = 150$). Curves use distinct markers and line styles; error bars are paired query-bootstrap 95% CIs and the horizontal dashed line is zero.

0.180] versus $-0.033 [-0.127, 0.060]$, with paired difference $+0.140 [0.053, 0.227]$. Both paired CIs exclude zero ($n = 150$): false-drop causes significantly greater downstream degradation than false-keep at both intervention levels. In the matched OpenAI sweep, hard-drop gain declines as query-critical false-drop rises from 0 to 0.279 (Figure 2b).

Labeled calibration supports empirical policy selection. We treat each corruption level as a separate deployment environment and split the 204 queries once into 102 calibration and 102 held-out test queries, stratified 17/17 per scenario by a frozen hash. For each environment, the gate selects hard-drop only if its calibration accuracy exceeds flat, otherwise flat; choices are frozen before test evaluation. It selects hard-drop for $p = 0, .1, .2$ and flat for $p = .3, .4, .5$. Pooled across 612 held-out environment–query outcomes, accuracy is 0.655 versus 0.569 always-flat, 0.641 always-hard-drop, and 0.583 always-soft-retain. A test-informed oracle policy reaches 0.662, leaving gate regret 0.007 versus 0.093, 0.021, and 0.078 for the three fixed policies. The 612 outcomes reuse 102 test queries across six regimes; they are not 612 independent questions. Calibration and test are query-disjoint but share 38 relevant memory chains. This retrospective policy requires matched labeled outcomes and known environment identity; it is neither an unlabeled confidence gate nor a statistical risk guarantee. In 1,000 exploratory alternative splits, hard-drop is selected 59.5% of the time at $p = .3$, showing decision instability near the crossover (Appendix K).

7 Oracle Upper Bound

Model	Flat	Hard-drop	AssemblyGain
GPT-4o-mini	0.600	0.775	$+0.175 [+0.130, +0.221]$
Haiku 4.5	0.558	0.760	$+0.202 [+0.157, +0.247]$
Sonnet	0.583	0.817	$+0.234 [+0.185, +0.283]$

Table 2: Oracle resolver on VALIDITYMEM-530, $n = 530$, regressed with the frozen committed-answer parser. All paired 95% CIs exclude zero.

With gold labels, hard-drop consistently improves behavior, establishing the high-reliability upper bound. The oracle result is useful setup rather than the main deployment claim: it shows that valid labels have limited value when the generator receives unresolved contradictions, while Figure 2 identifies when trusting those labels becomes unsafe.

8 Format versus Filtering

`structured_nofilter` applies the same headers and bullets to every memory but drops nothing. On GPT-4o-mini, format-only Gain is $+0.006 [-0.019, +0.028]$ and filtering beyond that adds $+0.170 [0.126, 0.215]$. On Haiku, the corresponding values are $-0.023 [-0.051, +0.004]$ and $+0.225 [0.177, 0.270]$. Formatting CIs include zero; the oracle benefit comes from filtering.

9 Published-Benchmark Diagnostic: MemoryAgentBench

VALIDITYMEM-530 is synthetic and self-authored. We therefore examine the same mechanism on

Model	Hop	Extraction architecture	Flat	Struct.	AssemblyGain (CI)
GPT-4o-mini	single	one-shot	0.360	0.800	+0.440 [+0.330, +0.550]
GPT-4o-mini	multi	one-shot	0.090	0.040	−0.050 [−0.100, −0.010]
GPT-4o-mini	single	hop-decomp.	0.370	0.780	+0.410 [+0.300, +0.510]
GPT-4o-mini	multi	hop-decomp.	0.140	0.280	+0.140 [+0.070, +0.210]
Claude Haiku 4.5	single	hop-decomp.	0.400	0.680	+0.280 [+0.190, +0.380]
Claude Haiku 4.5	multi	hop-decomp.	0.090	0.220	+0.130 [+0.060, +0.210]

Table 3: Tuned diagnostic on the published MemoryAgentBench FactConsolidation benchmark, $n = 100$ per row. The extraction prompt was refined after inspecting GPT-4o-mini failures on this evaluated sample; these are not held-out external-validation estimates.

MemoryAgentBench’s published FactConsolidation task (Hu et al., 2025), which tests selective forgetting through counterfactual edits to numbered facts. A higher serial number indicates a later re-statement, making max-serial resolution correct *conditional on accurate extraction*. We first use its 6k-token single- and multi-hop splits. Crucially, the extraction prompt was iteratively refined after diagnosing GPT-4o-mini failures on this same evaluated sample. We therefore label Table 3 a *tuned diagnostic*, not held-out validation.

Single-hop shows the predicted positive pattern. Extraction is essentially perfect here (99/100 GPT-4o-mini questions extract exactly the one group a single-hop question needs), and hard-drop wins by a wide margin on both models. Because prompt development touched the evaluated benchmark, we treat even this clean mechanism slice as diagnostic rather than confirmatory.

Multi-hop initially reversed the effect; we show why with a targeted intervention, not just observation. With one-shot extraction, only 34/100 GPT-4o-mini multi-hop questions extracted the two fact-groups needed; most collapsed distinct relations into one group. When extraction merges facts from different relations, max-serial marks all but the highest-numbered member “superseded,” and structured assembly *deletes* them – even when a deleted fact belonged to a different, unresolved hop. Structured was worse than flat here (one-shot row, Table 3): dropping content on an unreliable judgment is riskier than keeping everything.

We diagnosed this mechanism, hypothesized that fixing extraction would remove the reversal, and tested it: hop-decomposition extraction (decompose into sub-questions, extract per-hop, carry the prior hop’s fact forward as a hint), re-ran, and the sign flipped as predicted (hop-decomp. row). This is interventional evidence – diagnose, modify, confirm – not a purely correlational observation; we do not claim a fully randomized manipulation here (§6

provides that). The same hop-decomp. procedure is applied unchanged to every hop-decomp. condition in Table 3.

Frozen, question-disjoint replication is inconclusive. After freezing both prompts, we sampled 50 single-hop and 50 multi-hop questions by seeded SHA-256 rank from the previously untouched 32K-token split, excluding every normalized question duplicated in the tuned 6K-token sample. On GPT-4o-mini, answer-bearing extraction recall is 0.26 and query-critical false-drop incidence is 0.01; flat accuracy is 0.20 and hard-drop is 0.23, a paired Gain of +0.03 [−0.01, +0.08]. Extraction recall measures whether required answer evidence entered the resolver, whereas false-drop measures whether evidence reaching assembly was then deleted; upstream extraction misses are therefore reported separately. The 0.01 is the all-query incidence (1/100); conditional on answer evidence being extracted it is 1/26 (0.038). Multi-hop extraction recall is only 0.12. The direction is consistent with the tuned result but the interval includes zero, so this is a clean negative/inconclusive replication rather than external validation. Per-hop results and released provenance appear in Appendix F.

This is mechanistic corroboration, not independent replication. With the corrected grader, Claude’s multi-hop Gain is +0.130, close to GPT-4o-mini’s +0.140, despite 22% zero-extraction and under-extraction (mean 1.64 groups versus GPT-4o-mini’s 4.06). The shared prompt was tuned on GPT-4o-mini and Claude lacks a one-shot row, so cross-model differences remain confounded. The randomized corruption sweep, not this diagnostic, is the load-bearing test of the conditional-reliability claim.

10 Discussion

Three probes converge: a token-budget selector drops the current fact under scarcity (Appendix A);

label corruption drives AssemblyGain through zero, especially under false-drop (§6); and unreliable extraction deletes facts that were never stale (§9). Hard-drop converts available but ambiguous evidence into absence for the current generation call. Soft-retain removes that deletion channel but can still harm behavior by presenting incorrect validity labels as guidance. Systems should treat context filtering as a cost-sensitive decision and validate query-critical false-drop risk on matched labeled data. Label-free retention is an answer-accuracy fallback under the tested errors; it is not a general privacy or safety recommendation. Observed ResolutionAcc crossovers are model- and task-specific diagnostics, not universal deployment thresholds.

11 Conclusion

Validity resolution and downstream behavior are not the same quantity. Hard-drop improves accuracy with oracle labels but reverses sign as resolver errors accumulate. The asymmetric intervention identifies the mechanism: query-critical false drops are costlier than false keeps, while soft retention can still fail through misleading-label steering. On one synthetic sweep, labeled calibration approaches the test-informed oracle, but policy choice varies with the calibration sample. The frozen MemoryAgentBench replication is inconclusive. These results concern query-time filtering in question answering; natural resolver errors and long-horizon agent behavior require further evaluation.

Limitations

Three models from two providers, not a broad sweep. GPT-4o-mini and Claude Haiku 4.5 are the two smaller models tested; the larger Claude Sonnet model shows the largest effect, but other frontier-scale models remain untested. The matched soft-retention sweep was conducted only on GPT-4o-mini, so cross-provider replication remains future work.

Synthetic benchmark for the richer taxonomy. `expired_interval`, `historical_only`, and `non_monotonic_reversion` are VALIDITYMEM-530-only. The tuned MemoryAgentBench diagnostic covers the simpler current-vs-superseded case. Its frozen question-disjoint 32K-token replication is inconclusive and currently limited to GPT-4o-mini; independent cross-provider confirmation is still required.

Only one end-to-end memory-system sanity check. §2 reports a small Mem0 baseline; we do not re-implement MemGPT or A-Mem. Our claim concerns a variable these systems do not isolate, not end-to-end system outperformance.

Validity is not query relevance. Our harness retains historical facts only when the question type itself is historical; a more general policy conditioning what to drop on query intent, not just validity labels, is untested here.

Automatic grading retains a small false-positive mode. The blinded audit reported in §5 found two automated passes rejected by the annotator. Because its packet is balanced rather than prevalence-weighted, subgroup rates are diagnostic.

Conditional uncertainty and shared evidence. The reported query bootstrap treats related questions as independent despite memory-chain reuse; intervals may therefore be too narrow. Limited corruption realizations do not support across-seed uncertainty estimates. Repeating label corruption and generation with more seeds, and evaluating structured natural resolver errors, remain necessary. The alternative calibration splits reuse the same saved outputs and do not remedy these limitations.

Calibration is retrospective and label-dependent. The original split is frozen at the query level, not the memory-chain level, and evaluates one model under controlled corruption with labeled calibration outcomes. Shared chains, calibration-sample sensitivity, and dependence across regimes limit generalization. Prospective validation on disjoint trajectories under naturally occurring resolver errors remains necessary.

The multi-hop extraction fix is imperfect. Hop-decomposition fixed under-extraction but overshot into over-extraction (mean 4.06 groups vs. target ~ 2 , GPT-4o-mini); the cross-model prompt-tuning confound is discussed in §9. Nor is the calibrated policy an unlabeled confidence gate: its behavioral validation outcomes require deployment-matched labels, while global ResolutionAcc does not capture error cost.

Ethical Considerations

This work uses synthetic data (VALIDITYMEM-530, templated, no real personal data) and a published benchmark (MemoryAgentBench); we collect, use, or release no data involving real individuals. Our answer-accuracy evaluation does not

assess privacy, consent, or compliance with data-retention requirements. In deployed memory systems, incorrect retention may preserve outdated or sensitive user information, while premature deletion may remove context needed for safe personalization; accuracy-oriented retention must remain subordinate to applicable privacy, consent, and erasure requirements. Query-time omission is not persistent data deletion.

Artifact Availability

The supplementary archive accompanying this paper on [OpenReview \(submission 52\)](#) contains code, the generated dataset, prompts, frozen outputs, grading and audit reports, and the camera-ready calibration-sensitivity analysis. Its root `README.md` provides offline reproduction commands. Appendix L specifies context assembly exactly.

References

- Seyed Moein Abtahi, Rasa Rahnema, Hetkumar Patel, Neel Patel, Majid Fekri, and Tara Khani. 2026. Memanto: Typed semantic memory with information-theoretic retrieval for long-horizon agents. *arXiv preprint arXiv:2604.22085*.
- Hanxiang Chao, Yihan Bai, Rui Sheng, Tianle Li, and Yushi Sun. 2026. Stale: Can llm agents know when their memories are no longer valid? *arXiv preprint arXiv:2605.06527*.
- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.
- Yuanzhe Hu, Yu Wang, and Julian McAuley. 2025. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*.
- Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*.
- Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, Chengbao Liu, and Jie Tan. 2026. Timem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. *arXiv preprint arXiv:2601.02845*. Accepted to ACL 2026 Findings.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv: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 simulators of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.
- Vikas Reddy and Sumanth Challaram. 2026. Don’t ask the llm to track freshness: A deterministic recipe for memory conflict resolution. *arXiv preprint arXiv:2606.01435*.
- 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 (NeurIPS)*.
- Zhen Tao, Jinxiang Zhao, Peng Liu, Dinghao Xi, Yanfang Chen, Wei Xu, and Zhiyu Li. 2026. Memconflict: Evaluating long-term memory systems under memory conflicts. *arXiv preprint arXiv:2605.20926*.
- Sergey Volkov, Yang Li, and Ye Luo. 2026. Statefuse: Deterministic conflict-preserving memory for multi-agent systems. *arXiv preprint arXiv:2607.05844*.
- Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.
- Dongxu Yang. 2026. Control-plane placement shapes forgetting: An architectural study of agent memory across thirteen system configurations. *arXiv preprint arXiv:2606.15903*.
- Tiancheng Yang, Matthias Schonlau, and Ilia Sucholutsky. 2026. Selective qa over conflicting multi-source personal memory: A diagnostic testbed and method comparison. *arXiv preprint arXiv:2605.30087*.
- Chenghao Zhu, Nuo Chen, Yufei Gao, Yunyi Zhang, Prayag Tiwari, and Benyou Wang. 2025. Is your LLM outdated? a deep look at temporal generalization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7433–7457. Association for Computational Linguistics.

A Budget-Constrained Assembly

Real deployments cannot give a resolver unlimited context budget. We sweep `structured_budget`’s word budget over {15, 20, 30, 40, 60, 90, 150} (oracle resolver, GPT-4o-mini, $n = 102$). Regraded subset baselines are flat = 0.598 and unconstrained hard-drop = 0.804.

The curve is monotonic, crosses flat between 40 and 60 words, and reaches the unconstrained 0.804 ceiling at 90 words. Accuracy closely tracks

Budget	Acc.	Curr. recall	Avg. words
15	0.137	0.157	14.7
20	0.255	0.304	20.3
30	0.333	0.402	30.2
40	0.480	0.520	40.2
60	0.745	0.824	57.1
90	0.804	0.833	59.0
150	0.804	0.833	59.0

Table 4: Budget sweep, oracle resolver, structured_budget, $n = 102$ (same sub-sample as the subset baselines above).

current-fact recall. The failure mode is therefore query-critical evidence loss under scarcity, parallel to false-drop under label corruption.

B Soft-Retention Mechanism Metrics

p	ResAcc	NC P/R	Soft gain	Hard gain	CR/FD
0.0	1.000	1.000/1.000	+0.157	+0.181	1.000/0.000
0.1	0.894	0.950/0.925	+0.093	+0.108	0.946/0.054
0.2	0.832	0.916/0.916	+0.054	+0.074	0.907/0.093
0.3	0.737	0.846/0.827	−0.020	+0.015	0.833/0.167
0.4	0.612	0.769/0.796	−0.118	−0.098	0.745/0.255
0.5	0.547	0.722/0.677	−0.118	−0.123	0.721/0.279

Table 5: Matched GPT-4o-mini sweep ($n = 204$ per point). NC P/R is micro non-current precision/recall; gains are versus cached flat accuracy 0.623; Hard CR/FD is current-fact recall/query-critical false-drop rate. Soft-retain CR is 1.0 and FD is 0 throughout.

C Sensitivity-Subset Composition

Every scenario contributes exactly 34 queries to the $n = 204$ corruption subset. Because selection used the first queries in frozen, non-randomized file order, the subset is scenario-balanced rather than representative. It covers all five question types but overrepresents conflict probes and underrepresents yes/no and aggregation questions relative to the full benchmark.

Question type	Subset n	Subset %	Full n	Full %
extraction	105	51.5	270	50.9
yes/no	41	20.1	150	28.3
historical	17	8.3	40	7.5
probe	34	16.7	36	6.8
aggregation	7	3.4	34	6.4

Table 6: Question-type composition of the scenario-balanced sensitivity subset versus VALIDITYMEM-530.

D Calibration-Based Policy Evaluation

The same scenario-stratified split is reused across all six corruption regimes, preventing query overlap between calibration and test while allowing shared memory chains. Within each regime, the gate uses only calibration correctness labels. The oracle is a descriptive test-informed upper bound over flat, hard-drop, and soft-retain; it selects the same actions as an oracle restricted to flat versus hard-drop.

Policy	Test accuracy	Regret
Always flat	0.569	0.093
Always hard-drop	0.641	0.021
Always soft-retain	0.583	0.078
Calibration-selected	0.655	0.007
Oracle selection	0.662	0.000

Table 7: Pooled performance over 612 held-out regime-query outcomes. Regret is accuracy loss relative to oracle policy selection.

p	Cal. gain	Test flat	Test hard	Gate	Oracle
0.0	+0.137	0.569	0.794	hard	hard
0.1	+0.059	0.569	0.725	hard	hard
0.2	+0.010	0.569	0.706	hard	hard
0.3	−0.010	0.569	0.608	flat	hard
0.4	−0.176	0.569	0.549	flat	flat
0.5	−0.137	0.569	0.461	flat	flat

Table 8: Frozen decisions by corruption regime. Cal. gain is calibration hard-drop minus flat accuracy; ties and negative values select flat.

E Scenario-Level Sensitivity

At perfect labels, hard-drop is non-harmful in all six scenarios and helps most on non-monotonic reversion. At the largest corruption level, it becomes harmful in five scenarios; reversion remains positive. Thus scenario type alone is not a safety certificate: the same monotonic-supersession category changes from +0.059 to −0.235 as resolver errors increase.

F Held-Out MemoryAgentBench Detail

Table 10 gives the per-hop results for the frozen question-disjoint replication discussed in §9, separating extraction misses from subsequent context omissions.

G Resolver Reliability Tiers

A discrete, 4-tier complement to §6’s controlled sensitivity intervention.

Scenario	Flat	Hard _{p=0}	Gain	Hard _{p=.5}	Gain
current	1.000	1.000	0.000	0.706	−0.294
expired	0.382	0.647	+0.265	0.324	−0.059
historical	0.882	0.971	+0.088	0.588	−0.294
implicit	0.294	0.559	+0.265	0.265	−0.029
reversion	0.471	0.882	+0.412	0.647	+0.176
superseded	0.706	0.765	+0.059	0.471	−0.235

Table 9: Matched GPT-4o-mini sensitivity subset ($n = 34$ per scenario). Flat is invariant across corruption; hard-drop is shown at perfect labels and the largest corruption intervention.

Hop	n	Extract	FD	Flat/Hard	Gain [95% CI]
single	50	0.40	0.00	0.32/0.38	+0.06 [−0.02, 0.14]
multi	50	0.12	0.02	0.08/0.08	0.00 [0.00, 0.00]
overall	100	0.26	0.01	0.20/0.23	+0.03 [−0.01, 0.08]

Table 10: Frozen question-disjoint 32K-token GPT-4o-mini replication. Extract is answer-bearing extraction recall; FD is all-query query-critical false-drop incidence. Conditional FD among extracted-evidence cases is 0/20 (single), 1/6 (multi), and 1/26 (overall). Gain CIs use paired query bootstrap.

What deterministic and hybrid actually do. `deterministic` orders same-(entity, slot) memories chronologically from surface time expressions and exact-text dedup, then infers interval expiry from expiry phrases – no LLM calls, the harness’s analogue of a max-timestamp resolution recipe. Its main failure is mechanical: exact-text dedup treats a literal restatement as a no-op, so a `non_monotonic_reversion` chain (A, then B, then A again) collapses the final reassertion into the first mention. `hybrid` escalates any chain the dedup step flags as ambiguous to an LLM adjudicator, which sees the un-deduplicated candidates and names which is current – exactly the case designed to defeat `deterministic`’s shortcut (prompts in Appendix I).

Table 11 shows the same qualitative ordering without treating heterogeneous resolver tiers as a calibrated axis. GPT-4o-mini’s non-oracle gains are 0–0.009; Haiku’s rise from +0.013 to +0.096 as the tier improves; and Sonnet’s higher-reliability hybrid reaches +0.142. Oracle is largest for every model. The controlled corruption sweep in Figure 2, rather than this heterogeneous grid, supports the monotonic claim.

H Grading Robustness and Manual Audit

The first grader used raw substring matching; a second used token boundaries but still accepted a

Resolver	ResAcc	Flat/Struct. BehAcc	Gain
<i>GPT-4o-mini</i>			
no_labels	0.474	0.600 / 0.600	0.000
determ.	0.812	0.600 / 0.600	0.000
hybrid	0.799	0.600 / 0.609	+0.009
oracle	1.000	0.600 / 0.774	+0.174
<i>Claude Haiku 4.5</i>			
no_labels	0.474	0.557 / 0.570	+0.013
determ.	0.812	0.557 / 0.570	+0.013
hybrid	0.857	0.557 / 0.653	+0.096
oracle	1.000	0.557 / 0.758	+0.202
<i>Claude Sonnet</i>			
no_labels	0.474	0.587 / 0.594	+0.008
determ.	0.812	0.587 / 0.613	+0.026
hybrid	0.897	0.587 / 0.728	+0.142
oracle	1.000	0.587 / 0.823	+0.236

Table 11: Regraded AssemblyGain across heterogeneous resolver tiers. Oracle is largest for every model, but global accuracy does not form a calibrated deployment axis.

gold value mentioned after a leading UNKNOWN. We now freeze a committed-answer grader with three explicit rules: (1) leading UNKNOWN or an equivalent abstention always fails; (2) yes/no is determined by the first committed yes/no token, never later substring occurrence; and (3) categorical surface variants are normalized through a released, task-defined alias map (e.g., *Office* and *in-office* for *working from the office*). Regression tests cover abstention leakage, yes/no reversal, embedded numerals, punctuation, and aliases.

We regressed every stored output used in the manuscript without making new model calls. All three oracle headline CIs remain above zero. All three reliability curves still move from significantly positive at high reliability to significantly negative at low reliability and remain monotonic as reliability degrades. One near-zero Sonnet point changes sign (−0.005 → +0.005), shifting its crossover bracket by one grid interval; the reversal survives. MemoryAgentBench and budget numbers were also recomputed. Full machine-readable pre/post provenance is released in `reggraded_results.json`.

The earlier 80-item audit discovered the remaining bugs and therefore cannot validate the grader that those findings helped define. We replaced it with a new blinded 120-item packet: 40 responses per model, 20 per scenario, and 60 per assembly condition, with model, condition, scenario, and automated verdict hidden. The annotator completed all labels before the private key was opened. Agree-

ment is 118/120 (98.3%, Wilson 95% CI [94.1, 99.5]), with Cohen’s $\kappa = 0.962$ and confusion counts TP=80, FP=2, FN=0, TN=38 when the human label is the reference. Agreement is 100% for GPT-4o-mini, Sonnet, and hard-drop responses, 95.0% for Haiku, and 96.7% for flat responses. Both disagreements are flat Haiku outputs that mention the gold value without committing to it (an alternative answer and a future state). We do not tune the frozen grader on these audited cases; within the audit, both residual errors favor flat rather than hard-drop, so they do not suggest an inflated AssemblyGain. The completed packet and aggregate report are released.

I Prompts and Reproducibility Detail

The exact model identifiers are gpt-4o-mini, claude-haiku-4-5-20251001, and claude-sonnet-5; per-experiment API call counts are provided in the supplementary code release. The system prompts below preserve the wording used in the code (typeset quotation marks aside). The generation user template and assembly serialization are specified exactly in Appendix L; the remaining user turns here are abbreviated.

I.1 Generation prompt

Used by the Generator G for every VALIDITYMEM-530 and MemoryAgentBench answer, regardless of assembly mode.

System: “You are answering a question using ONLY the memory context provided. Do not use outside knowledge. If the context does not contain the answer, reply exactly: UNKNOWN. Answer as briefly as possible – a single value, name, number, or Yes/No – with no explanation.”

User: Memory context: {assembled context} Question: {query text} Answer:

I.2 Hybrid adjudication prompt

Used by the hybrid resolver tier (§G) to escalate chains its deterministic dedup step flags as ambiguous. Sees only surface text and surface time expressions – never the ground-truth timestamp.

System: “You are tracking which of several statements about the same fact is still true as of a given date. Read the statements in the order given (that is NOT necessarily chronological order). Reply with ONLY the number of the statement that is current as of the given date – nothing else.”

User: As of {query_time}:
1. {statement} (mentioned:
{surface_time_expr}) 2. ...
Which statement number is
current?

I.3 MemoryAgentBench extraction prompt

Used to extract candidate fact groups from MAB’s numbered fact list (§9). The one-shot version is used directly; the hop-decomp. version first decomposes the question (a separate call, prompt below), then runs this same extraction prompt once per sub-question, carrying the previous hop’s resolved fact forward as a hint.

Extraction system: “You extract facts relevant to a question from a numbered list of facts. Facts are numbered starting at 0. Some facts have been edited: a LATER (higher-numbered) fact about the exact same subject and relation OVERRIDES an earlier fact with the same subject and relation but a different value. Find every fact relevant to answering the question – including BOTH the original and any edited/overriding versions of it – and group them: each group is a list of facts about the exact same (subject, relation) pair. For multi-hop questions, include a separate group for EVERY hop needed to answer the question. Return ONLY strict JSON: a list of groups, each group a list of objects with “number” (int) and “text” (string, copied exactly from the fact list). No prose, no markdown fences, no explanation.”

Decomposition system (hop-decomp. only): “You decompose a multi-hop question into an ordered sequence of atomic single-hop sub-questions. Each sub-question after the first should be answerable only once the previous hop’s answer is known. Return ONLY strict JSON: a list of strings, each one sub-question, in order.”

J ValidityMem-530 Scenario Summary

K Exploratory Calibration-Split Sensitivity

To address calibration-sample dependence, we reanalyzed the saved GPT-4o-mini three-policy sweep after review using 1,000 alternative splits (seeds 0–999). Each uses the original SHA-256 ranking rule, 17 calibration and 17 test queries per scenario, and the same query assignment across all six regimes. The selection rule remains hard-drop iff its calibration accuracy strictly exceeds flat. No model outputs, corruption labels, grading rules, or

Scenario	Example question	<i>n</i>	Confl.?
current	“What is Priya’s current city of residence?”	40	No
Superseded	“What is Jamie’s current work mode?”	154	Yes
Expired interval	“What is Hiro’s work mode right now?”	120	Yes
Historical	“What was Sam’s city of residence back in 2024?”	80	Yes
Reversion	“What is Jordan’s current employer?”	100	Yes
Implicit conflict	“Would Tariq be comfortable ordering a steak dinner tonight?”	36	Yes

Table 12: VALIDITYMEM-530 scenario types, one generated example per type, and query count. Historical-only has 40 present-state extraction and 40 historically scoped queries; the example shows the latter.

Corruption <i>p</i>	Hard-drop selections
0.0	1000/1000 (100.0%)
0.1	998/1000 (99.8%)
0.2	958/1000 (95.8%)
0.3	595/1000 (59.5%)
0.4	5/1000 (0.5%)
0.5	1/1000 (0.1%)

Table 13: Empirical policy-choice frequencies over alternative calibration samples from one saved sweep. Near $p = .3$, the chosen action varies substantially.

policy rules were changed, and no additional model calls were made. We retain the original frozen split as the headline result rather than selecting a favorable alternative.

Mean selected-policy test accuracy is 0.680 across alternative splits, with empirical 2.5th–97.5th percentiles [0.637, 0.722]. Mean regret against each split’s test-informed oracle is 0.009, with corresponding percentiles [0.000, 0.038]. These describe sensitivity to repartitioning this finite saved dataset; overlapping splits are not independent experiments and these percentiles are not confidence intervals. The original split’s accuracy of 0.655 is retained in the main results.

The original calibration and test partitions each contain 83 distinct relevant memory chains, of which 38 occur in both partitions. Thus query disjointness does not imply independence of underlying evidence. Repartitioning queries does not correct this overlap, model-sampling uncertainty, or the limited number of corruption realizations. A prospective study should split at the trajectory or memory-chain level before collecting labeled calibration and test outcomes. The

script `scripts/analyze_realm_camera_ready.py` reproduces this diagnostic, including the overlap counts, and the full split-level results are in `calibration_sensitivity.json`.

L Exact Context Assembly Templates

The following serialization rules reproduce the non-budgeted implementations in `harness/assembly.py`. They distinguish the label-aware soft-retain condition from the label-free format-only ablation. Braced fields below are substituted literally; no quotation marks surround substituted text.

L.1 Generator user turn

All conditions use the following exact user-turn layout, with a single blank line before the question and no trailing newline after `Answer::`

```
Memory context:
{context_text}

Question: {query_text}
Answer:
```

The system prompt is the generation prompt in Appendix I. No additional assembly-specific system instruction is supplied.

L.2 Flat and format-only

Flat. Join every candidate’s `text` field with one newline, in the fixed input order. Add no header, bullet, validity label, or timestamp field.

Format-only. Emit the literal header `### All Facts`, a newline, then every candidate as `- {text}`, separated by single newlines in input order. Empty input produces the empty string. This condition ignores validity labels.

L.3 Hard-drop

Partition candidates by predicted validity. Emit the following nonempty sections in this exact order:

1. `### Current Facts: items predicted current.`
2. `### Flagged / Uncertain: items predicted uncertain or conflicting.`
3. `### Historical: items predicted historical, only if the query type is historical.`

Each section is its header, one newline, and one `- {text}` line per item. Join sections with two

newlines. Omit empty sections and do not append a final newline. Missing labels default to `uncertain`. Predicted superseded and expired items, and historical items for other queries, are omitted from this context. Relative input order is preserved within each section; no chronological sorting is added.

L.4 Soft-retain

Emit all candidates in three nonempty sections, preserving relative input order within each:

1. `### Current Facts: - {text}` for predicted current items.
2. `### Flagged / Uncertain:`
`- [UNCERTAIN] {text} or -`
`[CONFLICTING] {text}.`
3. `### Resolved Non-current`
`(retained; do not treat as`
`current): - [{LABEL}] {text},`
with the predicted label uppercased. For a historical query, historical items instead use
`- [HISTORICAL; QUERY-RELEVANT]`
`{text}.`

The long third header is one literal line in the generated prompt; any wrapping here is typesetting only. Newline and empty-section rules match hard-drop. The soft-retain sweep imposes no budget, so all candidate text is retained.

L.5 Ordering and budget details

For `VALIDITYMEM-530`, the harness concatenates relevant candidates and distractors and applies its query-keyed seeded shuffle once before assembly; every compared policy receives that same order. The released `seeded_shuffle` uses the integer represented by the SHA-256 hexadecimal digest of the UTF-8 query ID as its random seed. Section grouping is the only subsequent ordering change in hard-drop and soft-retain. `MemoryAgentBench` uses the fixed extracted candidate order supplied by its adapter; it does not use this synthetic-benchmark shuffle.

Budgeted hard-drop uses the same eligible buckets and order. Its budget counts whitespace-separated words in item lines, including each bullet as one word; headers are excluded from budget accounting. Within a bucket, the first line that exceeds the remaining allowance stops selection from that bucket. The next bucket is still considered with

the unused allowance. No item is truncated. Consequently, the complete serialized prompt can exceed the nominal word budget because headers also consume words. This is a word-count approximation, not a model-token budget.

The supplementary file `assembly_examples.txt` contains literal outputs produced by these implementations for one fixed illustrative input, including both historical-query settings. It is a serialization example, not an additional evaluated data point.