HABEAS: EVIDENCE-GATED REVERSIBLE MEMORY MANAGEMENT FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

External memory lets LLM agents reuse experience, but its value depends on which records remain available. History-based deletion permanently removes memories with low average utility on retrieved tasks. We identify two evidence gaps in this rule: deficient support under deterministic retrieval leaves many task–memory effects unidentified, and task outcomes confound a memory's contribution with difficulty, context, and execution noise. We propose HABEAS, an evidence-gated reversible lifecycle that separates suspension from verdict. Poor historical feedback quarantines a memory; paired audits re-execute completed tasks with and without it; restoration or deletion requires agreement across tasks; and restored memories remain under review. On two benchmarks (CICIoT2023 traffic classification and LAB-DB SQL tasks) and two models, HABEAS raises mean task success over history-based deletion in all sixteen model–benchmark–condition cells. In the primary CICIoT2023 Natural condition, gains are 6.13 and 2.67 percentage points, while the correct-memory deletion rate falls from up to 24.54% to zero. Ablations show that immediate quarantine accounts for most of the gain, the evidence gate reduces incorrect restorations relative to a simple reversible policy, and post-restoration review adds a smaller improvement. HABEAS creates the same-task evidence that retrieval logs cannot supply and uses it to govern irreversible memory decisions.

1 INTRODUCTION

External memory lets agents reuse reflections, cross-task insights, and reasoning strategies without updating model weights (Shinn et al., 2023; Zhao et al., 2024; Ouyang et al., 2026). Retrieved errors and mismatched experience can also impair subsequent behavior (Xiong et al., 2026). We call a memory good or bad according to whether it raises or lowers success on the task category it serves relative to a specified same-task alternative. This marginal-contribution view is also used by Wu et al. (2026), with label-free utility, and by Mishra & Mishra (2026) for a memory system as a whole: value depends on the memory–task pairing. Because this effect is not directly observable, our experiments follow Xiong et al. (2026) in using source correctness as a proxy. Correct/incorrect memories originate from correct/incorrect trajectories or labels (§5.4); good/bad describe their contextual effect. We study runtime management of a fixed bank. Memory reading selects stored records for the current task (Zhang et al., 2024); here, Top-k retrieval places the k highest-similarity records in context as demonstrations under a fixed ranking (Xiong et al., 2026). Only retrieved entries can affect the outcome and receive feedback.

History-based deletion removes an entry after sufficiently many retrievals if its average utility is low (Xiong et al., 2026). Two evidence gaps limit this score's interpretation (Figure 1). **Feedback coverage and support deficiency:** retrieval determines which task–memory pairs yield feedback. Missing feedback neither establishes uselessness nor triggers deletion automatically. **Outcome-based attribution:** managers assess memories from success or failure on retrieved tasks (Xiong et al., 2026; Zhang et al., 2026; Yang et al., 2026), although outcomes also reflect task difficulty, context, and execution noise. RoMeRL analyzes this attribution bias, while MemRL and CoEvo-Mem note credit-assignment difficulties among co-retrieved memories. The ambiguity persists with one retrieved entry: a useful memory may accompany failure on a hard task, and an easy task may succeed without its help. Deleting from this signal can therefore remove useful memories irreversibly.

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

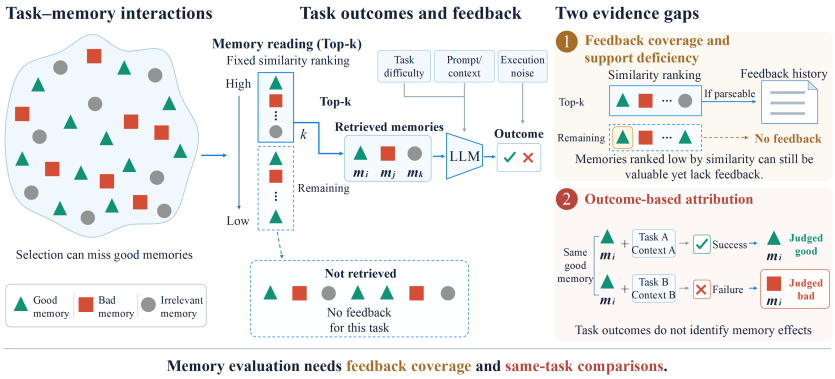


Figure 1: Two evidence gaps in memory evaluation.

Additional testing costs time and resources, but a suspicious entry need not remain in service while evidence is gathered. Classical sequential testing separates continued observation from acceptance and rejection (Wald, 1945). Inspired by this principle, HABEAS (Hold And Bring Evidence before Any Sentence) suspends service when risk appears, retains the entry for testing, and defers restoration or deletion until its evidence requirements are met. Paired executions compare the same completed task with and without the target under prescribed retrieval and backfill rules (§4.2, Appendix B.1). Restored memories remain subject to review.

On CICIoT2023 and LAB-DB with Qwen3.5-Flash and GPT-5.4-nano, HABEAS raises mean success over history-based deletion in all sixteen model–benchmark–condition cells. Gains in the primary CICIoT2023 Natural condition are 6.13 and 2.67 points. Across Natural conditions, correct-memory deletion falls from up to 24.54% to at most 0.78% (§6.1, Appendix C.4).

Our contributions are twofold. First, with memory content fixed and the retrieval rule given, we connect two feedback gaps to irreversible disposal. Proposition 1 formalizes non-identification under deficient support; Top-1 constructions show persistent non-selection under successful reuse and first-failure deletion of conditionally superior memories for any number of task classes (§3, Appendices A.3–A.4). Coverage diagnostics and forced-selection probes examine both gaps on two benchmarks. Second, HABEAS creates evidence through same-task paired audits and governs suspension, restoration, and deletion through a three-state lifecycle. Cross-task agreement gates permanent deletion, and post-restoration review keeps restoration revisable. Comparisons and ablations show how these mechanisms improve success while preserving correct memories.

2 Related Work

2.1 Experience Reuse and Feedback-Based Management

Experience reuse has grown from storing reflections to managing them by outcome feedback. Reflexion, ExpeL, and ReasoningBank store experience for later use (Shinn et al., 2023; Zhao et al., 2024; Ouyang et al., 2026); Xiong et al. (2026) study experience-following and selective addition/deletion, supplying our principal history-based deletion baseline. Among managers, Mem0 updates stored content by semantic relations (Chhikara et al., 2025), Memory-R1 learns management and utilization from outcome rewards (Yan et al., 2026), and MemRL, RoMeRL, and CoEvo-Mem update utility estimates or retrieval policies from feedback (Zhang et al., 2026; Yang et al., 2026; Ye et al., 2026). RoMeRL distinguishes observed from interventional marginal utility and analyzes the resulting attribution bias, CoEvo-Mem treats retrieval as determining which feedback a memory receives, and MemRL notes the difficulty of assigning credit among several retrieved memories; our study is complementary: with memory content fixed and the retrieval rule given, we ask how these gaps affect irreversible disposal. Methods that permanently delete on historical outcomes, such as history-based deletion, read a memory's value from the tasks on which it happened to be retrieved—an unretrieved memory is never assessed and one retrieved on hard tasks looks harmful (§3)—and cannot correct

the deletion afterwards. HABEAS creates the missing evidence with same-task paired audits and keeps every memory reversible until that evidence arrives.

2.2 Interventional auditing and persistent correction

A second line, memory security, audits stored memories against poisoning: it asks whether a record is malicious, whereas we ask whether an ordinary record helps or hurts on the tasks it serves, but both decide a memory's fate from indirect evidence. Memory attacks show that apparently ordinary records can alter behavior (Chen et al., 2024); A-MemGuard validates actions online against consensus and defensive memories (Wei et al., 2025); MemAudit ranks post-incident removals by counterfactual removal/retrieval and structural anomalies (Tan et al., 2026); and Hindsight Memory-PRM uses controlled removal, re-retrieval, and re-answering to derive utility supervision for training a manager whose actions are write, merge, and no-op over retained version chains, with destructive updates excluded (Jia et al., 2026). These interventions serve post-incident diagnosis, training-time supervision, or pre-action validation; HABEAS runs a persistent, budgeted runtime lifecycle in which paired evidence supports both restoration and deletion and restored memories remain under review.

3 Problem formulation and evidence gaps

We fix the object of study: a frozen agent A (its weights are never updated) serves tasks $x_1, \ldots, x_T$ from an initial bank $M_0 = \{m_1, \ldots, m_N\}$ whose contents are not extended, while entries change state. We call these service tasks, as opposed to the audit executions of §4. With M_t^{ACT} the entries active (eligible for retrieval) at time t, the retrieved context S_t, the answer y_t, and the binary feedback r_t returned by the task checker R are

$$S_t = \mathrm{TopK}_k(x_t; M_t^{\mathrm{ACT}}), \qquad y_t = A(x_t, S_t), \qquad r_t = R(x_t, y_t) \in \{0, 1\}. \tag{1}$$

Let $n_i(t)$ count the service retrievals of m_i and $\bar{u}_i(t)$ average their utilities $u_t = \Phi(r_t)$; history-based deletion (HM) removes m_i once $n_i \geq n_{\min}$ and $\bar{u}_i \leq \beta$. This statistic summarizes outcomes on the tasks where m_i was used, not the difference its removal would make.

3.1 Feedback coverage and support deficiency

Feedback is available only for task–memory pairs that enter the candidate set and are selected: the checker R of Eq. (1) scores the outcome of a selected pair, and no other pair produces feedback. Using the indicator $\mathbb{I}[\cdot]$, write $E_{i,t} = \mathbb{I}[m_i \in S_t]$ for selection. Under deterministic Top-k retrieval, selection is fixed given the task and the current active bank; across a task region U its probability depends on the task distribution and on the bank states reached during management. Zero support means that the logging policy never selects the target on U. This is the support problem of off-policy evaluation, which combines a reward model with a correction for the logging policy (Dudík et al., 2011) and in which deficient support limits what logged feedback identifies unless smoothness assumptions tighten partial-identification bounds outside overlap (Sachdeva et al., 2020; Khan et al., 2024). Proposition 1 states the memory-specific version: additional logs from the same policy cannot identify the sign of the target contrast when the admissible models contain observationally identical worlds with opposite signs.

Proposition 1 (non-identification under deficient support). Fix the observable information and a target region on which the logging policy never selects m_i. If the permitted outcome models allow two worlds that agree on all logged observations but assign opposite signs to the target's average contextual effect, that sign—and a retain/remove decision defined by it—is not point-identified from the logs. Appendix A.1 gives the construction and proof.

Proposition 2 (persistent coverage gap under successful reuse). Consider K isolated task classes with fixed within-class rankings, Top-1 retrieval, and a rule that retains the selected memory on success and deletes it on failure, where good memories always succeed and bad ones always fail. If class z holds $g_z \geq 1$ good memories, receives L_z tasks, and has its first good memory at rank R_z,

then with $g_{\text{tot}} = \sum_z g_z$ the number of distinct good memories never selected is exactly $U_{\text{cov}} = g_{\text{tot}} - \sum_{z=1}^{K} \mathbf{1}\{L_z \geq R_z\} = g_{\text{tot}} - K + \sum_{z=1}^{K} \mathbf{1}\{L_z < R_z\}$.

Once every class reaches its first good memory, $U_{\text{cov}} = g_{\text{tot}} - K$ regardless of further task counts; with g good memories per class the unselected fraction is $1 - 1/g$. Repeated success thus coexists with a positive coverage gap whenever some class holds several good memories (proof and extensions in Appendix A.3).

Two diagnostics measure the missing feedback. The coverage census freezes the banks, tasks, and ranking of both benchmarks used throughout (LAB-DB SQL tasks and CICIoT2023 traffic classification, hereafter LAB and CICIoT; §5.1) without any management, enumerates every potentially relevant task–memory pair, and follows it from candidate to selected (Table 1; protocol in Appendix A). Only 3.695% of potentially relevant pairs in LAB and 23.08% in CICIoT are ever selected, so the frozen retrieval logs contain no outcome feedback for the remaining 96.3%/76.9%.

Table 1: Coverage of potentially relevant memory–task pairs.

Benchmark / model	Pairs	Top-10 candidates	Selected
LAB / Qwen3.5-Flash	7,686	2,419 (31.47%)	284 (3.695%)
CICIoT / Qwen3.5-Flash	37,500	26,134 (69.69%)	8,655 (23.080%)
CICIoT / GPT-5.4-nano	37,500	26,134 (69.69%)	8,655 (23.080%)

Natural banks without management. Pairs: potentially relevant task–memory pairs (same top-level SQL operation for LAB, same traffic class for CICIoT), the denominator of all percentages; Selected: Top-1 for LAB, Top-3 for CICIoT (protocol in Appendix A). The CICIoT rows coincide because both models share the same banks, tasks, and frozen retrieval rankings.

A controlled support simulation confirms that more data closes the gap only under positive support: with utilities, tasks, noise, and seeds fixed and only the target's eligibility varied (full, narrow at 4% of tasks, or zero), the decision error of the doubly robust estimator (Dudík et al., 2011) at $n = 10{,}000$ is 0%/0%/50%, because under zero support two worlds with opposite target effects generate identical logs (Appendix A.1, Table A1). HABEAS therefore does not wait for the logs: it creates selection on purpose, temporarily including or excluding a flagged memory on a completed task (§4.2).

3.2 Outcome-based attribution

Even a selected memory can be misjudged if task success is attributed directly to its quality. Write the success probability of task x with memory m_i in context as $b(x) + \tau_i(x)$, where the solvability $b(x)$ is the probability that the agent solves x when m_i is absent (its slot filled by the prescribed replacement) and the contextual effect $\tau_i(x)$ is the change caused by placing m_i in the context. The observed mean decomposes as $\mu_i^{\text{obs}} = \mathbb{E}[b(x)|m_i \in S_t] + \mathbb{E}[\tau_i(x)|m_i \in S_t]$: it mixes the baseline difficulty of the tasks on which m_i happened to be retrieved with the effect that management wants to know. Because retrieval is by similarity, the tasks on which m_i is retrieved resemble its source task rather than forming a random sample (Xiong et al., 2026), so their baseline solvability can differ systematically from the average: a memory with $\tau_i > 0$ retrieved on hard tasks can have $\bar{u}_i \leq \beta$ and be removed by history-based deletion, while a memory with $\tau_i < 0$ retrieved on easy tasks can keep $\bar{u}_i > \beta$. Averaging more outcomes estimates the mixture more precisely without separating its two terms. The issue is the missing conditional contrast, not only how a shared reward is divided among co-retrieved entries.

Proposition 3 (first-failure deletion of conditionally superior memories). Under the rule of Proposition 2, let a good memory succeed independently with probability $p_{G,z} < 1$, greater than its same-class comparison probability $p_{B,z}$, and let W count its uses up to and including the first failure. Once in service, it is deleted within h class-specific opportunities with probability $\Pr(W \leq h) = 1 - p_{G,z}^{h} > 0$ for $h \geq 1$.

Appendix A.4 derives the exact minimum expected deletion count over all fixed rankings for arbitrary K—in the illustrative ten-class bank even the most favorable ordering yields 49.999832 expected good-memory deletions out of 50—and with continuing opportunities all good memories are eventually deleted almost surely: turning a stochastic failure into irreversible removal produces this loss even at fixed difficulty.

The evidence that disposition needs is $\tau_i(x)$ itself. For the same completed task x and current bank M, let S^+ denote the retrieval context that includes the target memory m_i (the memory under audit) under the audit eligibility rules of §4.2, and S^- the context that excludes it with prescribed backfill (the vacated slot is filled by the next-ranked eligible memory). We ask for the expected reward difference between these two specified contexts:

$$\delta_i(x; M) = \mathbb{E}\big[R(x, A(x, S^+)) - R(x, A(x, S^-))\big]. \tag{2}$$

The contrast fixes the task, agent, bank snapshot, and protocol; only the target and its prescribed replacement differ, so $b(x)$ cancels in expectation and $\delta_i(x; M) = \tau_i(x)$. A realized pair yields $\Delta = r^+ - r^- \in \{-1, 0, +1\}$ with $\mathbb{E}[\Delta] = \delta_i(x; M)$; ties carry no direction. Because one pair is one noisy sample, HABEAS requires agreement across distinct tasks and nearest-neighbor groups (§4.3; an illustrative sign-error calculation is in Appendix A.2) and evaluates the requirement through lifecycle ablations (§6.3).

The forced-selection probe shows this contrast in practice: memories the ranking omitted (raw rank above 10) were placed into the context of their own related tasks on the Natural banks and compared with the natural context on the same task (Appendix A.2). Of 37 LAB pairs, 3 changed from failure to success and 1 from success to failure; of 92 CICIoT pairs, 6 and 4; most pairs were ties, which is why one pair is not a verdict.

4 HABEAS: EVIDENCE-GATED REVERSIBLE MANAGEMENT

Figure 2 summarizes the pipeline: a memory serves until its task feedback turns suspicious, is then quarantined and audited by re-executing completed tasks with and without it, is restored or permanently deleted when paired evidence agrees across tasks, and, once restored, is reviewed by the same audit whenever new feedback casts doubt. Sections 4.1–4.4 describe the CICIoT instantiation; Appendix B.1 gives the LAB adaptation, which shares the trigger, evidence requirements, and review rules and differs in its Top-1 retrieval interface and in using no semantic ranking. Each memory holds one of three states, $s_i \in \{\text{ACT}, \text{QUA}, \text{DEL}\}$—active, quarantined (kept but withheld from retrieval), or deleted—and changes state only through these mechanisms; thresholds and caps are listed in Appendix B (Table A3).

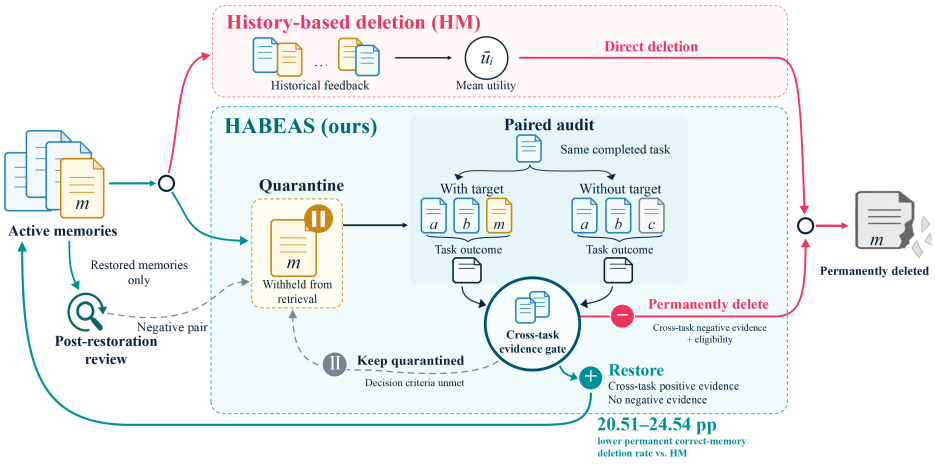


Figure 2: HABEAS memory-management framework. The displayed deletion-rate reductions refer to CICIoT2023 Natural (Table A11).

4.1 HISTORICAL FEEDBACK TRIGGERS QUARANTINE

HABEAS turns the statistic that HM uses for deletion into a trigger for a reversible state: a never-restored active entry whose use count and mean centered feedback (task feedback centered by the

frozen utility transform of Appendix B) cross the risk thresholds moves to QUA at the next checkpoint (every fixed number of service tasks; Appendix B), keeping its content while withholding it from retrieval. On both benchmarks the thresholds and checkpoints are those under which HM deletes, so quarantine takes effect exactly where HM would delete. The trigger flags risk only; paired executions of the same task with and without the target (§4.2) decide restoration or deletion. Audits are acquired for flagged entries that meet the eligibility rules, not for every unselected memory in the bank.

4.2 Paired audits compare specified contexts

HABEAS compares two executions of the same completed task x: S^+ temporarily admits the quarantined target and S^- excludes it, both following the same full-bank ranking, state filtering, and backfill rules, so the contexts differ only in the target and its prescribed replacement. A pair is eligible only if the two contexts differ and the fixed applicability checks pass. The checker R of Eq. (1) scores the pair as $\Delta = r^+ - r^- \in \{-1, 0, +1\}$; ties provide no directional evidence and invalid executions none. An exactly matching service response—same task, same ordered retrieval context, fixed execution settings—may supply one side, every tenth pair executes both sides afresh, and reuse does not depend on the answer's correctness; in the main CICIoT runs, 76.29% of the 2,269 pairs reused one side (Appendix D). The comparison remains subject to execution variability, so restoration and deletion require corroborating results across tasks (§4.3).

4.3 Evidence requirements for restoration and deletion

Within a quarantine episode e (from an entry's quarantine until its restoration or deletion), each valid outcome is recorded with its task ID and nearest-neighbor group $\rho(x)$, the nearest memory ID under the fixed, unfiltered ranking—a grouping by retrieval structure, not true classes. Restoration requires positive evidence from at least two distinct tasks and two distinct groups with no valid negative evidence in the episode:

$$\mathrm{Restore}(i, e) = \mathbf{1}\big[\,|\mathcal{T}^+_{i,e}| \geq 2 \;\wedge\; |\mathcal{R}^+_{i,e}| \geq 2 \;\wedge\; N^-_{i,e} = 0\,\big]. \tag{3}$$

Here $\mathcal{T}^+_{i,e}$ and $\mathcal{R}^+_{i,e}$ are the sets of distinct tasks and groups with a positive outcome for m_i in episode e, and $N^-_{i,e}$ counts valid negative outcomes. Permanent deletion requires negative comparisons on two tasks and two groups plus the fixed deletion eligibility checks (Appendix B); insufficient, tied, or invalid evidence leaves the entry in quarantine. Neither a task failure nor one negative probe deletes, and one positive pair does not restore (evaluated in §6.3).

4.4 Restoration remains revisable

A restored memory regains global retrieval eligibility and receives a separate post-restoration review history: enough new uses, or fewer uses with low task-level feedback, trigger a review by an eligible paired audit. A valid negative comparison returns it to quarantine, positive evidence retains it, and ties or invalid pairs add no support, so a mistaken restoration is reversed by evidence while a good restored memory is not re-quarantined by confounded feedback (§6.3).

4.5 Evidence allocation and budget

All audit stages—restoration audits and post-restoration reviews—draw on one bounded budget of paired executions and model calls per service task. A semantic ranking (an LLM call that orders eligible quarantined candidates by relevance, distinct from the similarity ranking used for retrieval) and, where applicable, prediction disagreement (the total-variation distance between the two contexts' predicted label distributions) prioritize the candidates most likely to yield directional evidence; these signals only order tests, and state changes require audit outcomes. The allocation policy, its alternatives, and the cost accounting are in Appendix D.

5 EXPERIMENTAL DESIGN

5.1 BENCHMARKS AND MODELS

We evaluate on two benchmarks with different retrieval interfaces and two models. CICIoT2023 (Neto et al., 2023) is an eight-class network-traffic classification task with a common pool of 1,000 tasks, 100 initial memories per seed, and Top-3 retrieval; LAB-DB adapts LifelongAgentBench SQL tasks (Zheng et al., 2025) with Top-1 retrieval and 300 tasks per run. CICIoT keeps the Top-3 configuration of prior memory-management work (Xiong et al., 2026); LAB uses a controlled Top-1 adaptation of its experience-replay framework to limit attribution confounding among multiple retrieved memories. Within each benchmark, all memory-management methods share the same retrieval configuration. Both are served by Qwen3.5-Flash (qwen3.5-flash-2026-02-23; Qwen for short) and GPT-5.4-nano (gpt-5.4-nano-2026-03-17; GPT for short), with three seeds per cell (a seed fixes the initial bank's source sampling, corruption positions, and task order). Natural, the primary condition, retains naturally incorrect memories from experience collection. Both benchmarks use the same Stress mechanism: transferable erroneous principles are injected into memory, with Stress10/20/40 targeting potential Top-1 selection to controlled errors on 10%/20%/40% of tasks (Appendix C.3). Benchmark details are in Appendix C.1.

5.2 COMPARED METHODS

Our principal baseline is history-based deletion, HM (Xiong et al., 2026), the representative existing management rule: it tracks each memory's use count n_i and its mean utility $\bar{u}_i$ over the tasks on which it was retrieved, and permanently deletes the memory once $n_i \geq n_{\min}$ and $\bar{u}_i \leq \beta$. HM and HABEAS share the fixed bank, task stream, retrieval rule, and service-feedback interface, with the thresholds of Appendix B. They also share the risk trigger and checkpoints on both benchmarks, so the two differ only in what they do once a memory's feedback turns poor. HABEAS is the complete method of §4. Three further references bracket the comparison: No-memory answers every task without retrieval; RAG (retrieval without management) retrieves the Top-k memories under the fixed ranking and never changes the bank; and Oracle serves the bank with incorrect memories removed in advance—a privileged reference for how much of the gap quality information would close, not a competing manager.

5.3 ABLATIONS AND CONTROLS

Three further comparisons isolate the lifecycle's parts, each answering one question. Lifecycle ablations ask what each stage contributes: w/o quarantine keeps flagged memories in service while their audits run (is immediate quarantine necessary?); Single-positive restoration restores on one positive pair instead of two tasks and two groups (is the evidence requirement necessary?); w/o post-restoration management removes review tests and reverts to the original shared-task-feedback quarantine rule (is a restored memory safe without review?). The post-freeze management-policy comparison (run after the method and its thresholds were frozen) asks whether the full lifecycle adds to the simplest reversible policy: on five new 1,000-task CICIoT blocks (disjoint task pools), Audit-Delete leaves flagged entries in service while awaiting evidence, Simple-Reversible quarantines immediately and restores or deletes on a net vote (positive minus negative audit outcomes reaching a threshold), and HABEAS is not retuned, with all three sharing the same audit eligibility rules and caps. The audit-policy comparison asks whether active evidence acquisition pays and what an allocator buys: B0 manages without audits, Uniform allocates audits evenly, and Targeted is the complete HABEAS configuration; it is reported in Appendix D. Ablation and control details are in Appendices D–F.

5.4 METRICS

Success is the share of service tasks solved, $\mathrm{Succ} = (1/T)\sum_t r_t$; audits and semantic ranking never enter it. Because management can fail in two directions—losing correct memories and serving incorrect ones—we report both. With G and B the initial sets of correct and incorrect memories (source labels, §1) and A_t, Q_t, D_t the active, quarantined, and deleted sets at time t, the correct-memory deletion rate D_G, the terminal and time-averaged correct-memory unavailability U_G and

$\bar{U}_G$, the terminal incorrect-memory availability A_B, and the retrieved-incorrect share E_B are defined in Appendix C.2 (Eqs. (A2)–(A3)). These metrics record what management does to source-labeled memories rather than each memory's effect, since a correct memory can still hurt on a mismatched task (§3.2); every rate is computed per seed or block before equal-weight averaging. Cost is reported as model calls and tokens per run, counting every call a method requires (Appendix D).

6 Results and analysis

6.1 Complete-method task performance

On CICIoT2023, HABEAS exceeds HM in mean success in all eight model–condition cells (Table 2): Natural gains are 6.13 points for Qwen and 2.67 for GPT; Stress gains 0.37–7.77 and 0.47–1.70. Per-seed paired differences range from −0.5 to +13.1 points (Table A8); the three seeds share a task pool, so the eight cells are not independent replicates.

Table 2: Success of the five compared methods on the frozen CICIoT2023 eight-class subset.

Model	Regime	No-memory	RAG	HM	HABEAS	Oracle*
Qwen	N	47.73 ± 1.36	57.43 ± 6.37	67.67 ± 2.10	73.80 ± 6.17	85.40 ± 5.97
Qwen	S10	47.73 ± 1.36	55.60 ± 5.09	67.90 ± 7.01	75.67 ± 7.33	85.60 ± 6.58
Qwen	S20	47.73 ± 1.36	56.00 ± 5.25	72.57 ± 12.05	72.93 ± 11.30	85.90 ± 5.96
Qwen	S40	47.73 ± 1.36	49.33 ± 5.78	65.67 ± 10.38	67.70 ± 6.70	85.90 ± 6.56
GPT	N	36.67 ± 0.92	39.53 ± 5.02	57.10 ± 5.48	59.77 ± 4.01	81.87 ± 1.70
GPT	S10	36.67 ± 0.92	38.93 ± 4.74	54.90 ± 4.08	56.60 ± 5.62	81.63 ± 2.44
GPT	S20	36.67 ± 0.92	39.50 ± 3.10	54.93 ± 2.50	56.63 ± 3.06	81.97 ± 1.70
GPT	S40	36.67 ± 0.92	37.13 ± 2.55	57.87 ± 3.33	58.33 ± 2.57	82.37 ± 1.71

Success rate (%), mean ± sample SD over three seeds; N = Natural, S10/S20/S40 = Stress10/20/40 (Appendix C.3). Qwen = Qwen3.5-Flash; GPT = GPT-5.4-nano. Methods as in §5.2; *Oracle is a privileged human-audited reference, not a manager. No-memory shares one reference across Stress conditions.

LAB illustrates a different regime (Table A6): injected errors make memory a liability, so the management question is how much of the damage is recovered. HABEAS's Stress-averaged success is 68.22%/80.33% for Qwen/GPT, against HM's 66.48%/78.93% and unmanaged memory's 67.04%/78.15%, and it exceeds HM in all eight cells. The No-memory reference (78.67%/88.89%) measures the damage the injected bank causes; among methods that consult memory, HABEAS's Stress-averaged success is the highest.

6.2 Increment beyond a simple reversible policy

The five-block comparison asks whether the full lifecycle adds anything to the simplest reversible policy (Table A20; gains are differences in equal-weight mean success over the five blocks). In Natural, HABEAS is 0.88 points above Simple-Reversible with a correct-memory deletion rate of zero against 2.98%; in Stress20 it is 2.96 points above, with time-averaged correct-memory unavailability and E_B lower by 1.04 and 0.54 points and deletion again zero against 6.11%. Both use the same number of measured pairs (96.6 and 96.8) while HABEAS consumes 4.32%/4.47% fewer tokens. Audit-Delete, which keeps flagged entries in service while waiting for evidence, reaches only 52.18%/52.48%, 15.1/15.7 points below HABEAS.

HABEAS records two incorrect-memory restorations per condition versus 6/20 under Simple-Reversible; complete outcomes, including invalid pairs, are in Appendix F.

6.3 Lifecycle ablations

Figure 3 compares the full method with the three ablated variants of §5.3 on CICIoT2023 Natural with Qwen3.5-Flash, reporting task success, incorrect-memory retrieval, incorrect selections after restoration, and re-quarantine of correct memories; Table A15 (Appendix E) gives the full metrics.

Removing immediate quarantine lowers Natural success from 73.80% to 57.57% while E_B rises from 24.67% to 46.70% (Table A15): its zero correct-memory unavailability comes from continu-

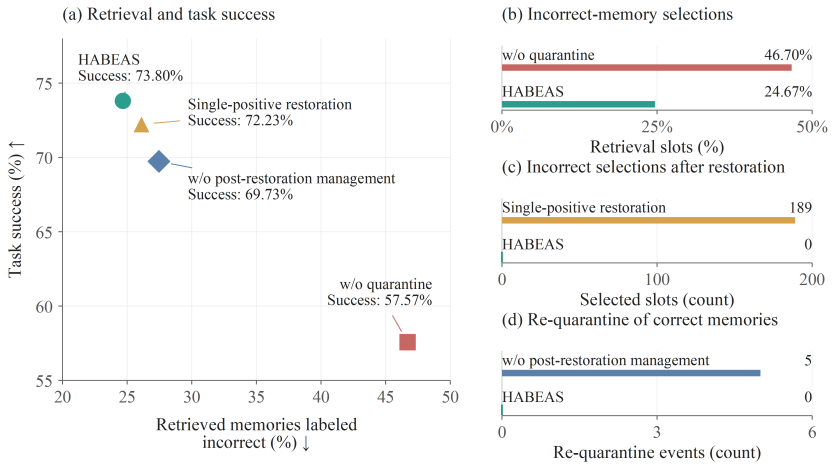


Figure 3: Ablation of lifecycle management mechanisms. The full HABEAS method is compared with ablated variants using Qwen3.5-Flash on CICIoT2023 Natural. Rates and success are averaged over three seeds; event counts are summed across the three seeds.

ing to serve risky entries—containment is what quarantine buys. Single-positive restoration raises terminal correct-memory availability but lowers mean success and admits three incorrect-memory restorations with 189 subsequent incorrect selections. Paired by seed, the difference against w/o quarantine is positive in all three seeds, whereas the smaller gains over the other two variants vary across seeds (Table A19).

Without post-restoration management, mean success is 69.73%, time-averaged correct-memory unavailability rises from 11.31% to 14.32%, and five correct memories are re-quarantined after restoration, versus zero under the full method; removing review tests and their feedback handling together lowers mean success by 4.07 points. All four variants keep a zero correct-memory deletion rate; the differences lie in availability and in which restorations survive.

7 Discussion and Limitations

Discussion of audit cost, evidence, and restoration behavior appears in Appendix H; limitations and future directions are detailed in Appendix I.

8 Conclusion

HABEAS separates the two evidence gaps of feedback-based memory management—limited coverage under deficient support and outcome-based attribution—from the irreversible decision they are used to justify: observing success or failure is not observing what a memory changed. Historical feedback triggers reversible containment, same-task paired comparisons inform restoration and deletion, and review keeps restored service answerable. Across two benchmarks and two models, HABEAS improves over history-based deletion in every cell and, in Natural, reduces the correct-memory deletion rate from up to 24.54% to at most 0.78%; weakening quarantine, the restoration requirement, or post-restoration management lowers mean success. Memory-management decisions should match evidence requirements to the consequences of an action rather than infer irreversible deletion from task outcomes.

AI USE STATEMENT

In this work, we used generative AI tools for the following purposes. (1) To aid and polish writing, including language editing, improving readability, and preparing the manuscript. (2) For retrieval and discovery, including searching for and identifying potentially relevant literature. (3) To draft sections of the paper, which were used as a basis for author revision. (4) To generate synthetic datasets used in this work. (5) To assist with code development, including writing and debugging research code. Details of these uses are described in the paper. The authors take responsibility for reviewing and validating AI-assisted contributions and for the final content of this work, including the text, claims, synthetic data, and code.

ETHICS STATEMENT

The evaluation uses benchmark classification and database tasks with controlled memory corruption. Incorrect management decisions can remove useful experience or retain harmful advice, and audits consume execution resources. Consequential deployment requires domain-specific validation and access controls.

REPRODUCIBILITY STATEMENT

Management rules, thresholds, budgets, and reuse rules are specified in §4 and Appendix B (Algorithm 1, Table A3, Appendix B.1). Benchmark construction, scoring, statistical units, and Stress injection are described in Appendix C. Per-seed task-success results and paired differences are reported in Tables A7, A8, and A19; block-level availability is reported in Table A23. Dated model versions are listed in §5.1. A project demo will be released with the final version of the paper.

REFERENCES

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2407.12784.

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

Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on Machine Learning*, 2011. URL https://arxiv.org/abs/1103.4601.

Haoxuan Jia, Yang Liu, Yingguang Yang, Yancheng Chen, Chongyang Zhang, Hao Zheng, Qian Li, Yulin Huang, Jianshen Zhang, Yongzhi Qi, Shang Luo, Kefu Xu, Hao Peng, Junyu Lu, Du Cheng, Philip S. Yu, and Bin Chong. Hindsight Memory-PRM: Supervising memory management with auditable hindsight credit. *arXiv preprint arXiv:2608.29605*, 2026. URL https://arxiv.org/abs/2608.29605.

Samir Khan, Martin Saveski, and Johan Ugander. Off-policy evaluation beyond overlap: Sharp partial identification under smoothness. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 23734–23757, 2024. URL https://proceedings.mlr.press/v235/khan24b.html.

Shweta Mishra and Shashank Mishra. When does memory help? A cost-aware evaluation of long-term memory in tool-using LLM agents. *arXiv preprint arXiv:2609.05441*, 2026. URL https://arxiv.org/abs/2609.05441.

Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A. Ghorbani. CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*, 23(13):5941, 2023. doi: 10.3390/s23135941. URL https://doi.org/10.3390/s23135941.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25140.

Noveen Sachdeva, Yi Su, and Thorsten Joachims. Off-policy bandits with deficient support. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020. doi: 10.1145/3394486.3403139. URL https://arxiv.org/abs/2006.09438.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Zhewen Tan, Yilun Yao, Huiyan Jin, Wenhan Yu, Guoan Wang, Mengyuan Fan, Liang Lu, Feng Liu, Xiangzheng Zhang, Duohe Ma, Tong Yang, and Lin Sun. MemAudit: Post-hoc auditing of poisoned agent memory via causal attribution and structural anomaly detection. *arXiv preprint arXiv:2605.23723*, 2026. URL https://arxiv.org/abs/2605.23723.

Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945. doi: 10.1214/aoms/1177731118. URL https://doi.org/10.1214/aoms/1177731118.

Qianshan Wei, Tengchao Yang, Yaochen Wang, Xinfeng Li, Lijun Li, Zhenfei Yin, Yi Zhan, Thorsten Holz, Zhiqiang Lin, and XiaoFeng Wang. A-MemGuard: A proactive defense framework for LLM-based agent memory. *arXiv preprint arXiv:2510.02373*, 2025. URL https://arxiv.org/abs/2510.02373.

Beining Wu, Zihao Ding, Jun Huang, and Yanxiao Zhao. Forget to improve: On-device LLM-agent continual learning via budget-curated memory. *arXiv preprint arXiv:2606.25115*, 2026. URL https://arxiv.org/abs/2606.25115.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts LLM agents: An empirical study of experience-following behavior. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 623–645, 2026. URL https://aclanthology.org/2026.acl-long.27/.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, 2026. URL https://aclanthology.org/2026.acl-long.583/.

Yi Yang, Zhennan Chen, Yihong Zhuang, Tiehan Fan, Yinan Chen, Jian Li, Jian Yang, and Ying Tai. RoMeRL: Balancing feedback coverage and the memory-reward trap in self-evolving agent memory via reduced-order utility states. *arXiv preprint arXiv:2608.02508*, 2026. URL https://arxiv.org/abs/2608.02508.

Bowen Ye, Yongchao Xu, Zhijian Li, Xiang Yin, Junkai Ma, and Wenzhao Li. CoEvo-Mem: Co-evolving retrieval policy and memory bank for LLM agents. *arXiv preprint arXiv:2608.01739*, 2026. URL https://arxiv.org/abs/2608.01739.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026. URL https://arxiv.org/abs/2601.03192.

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

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2308.10144.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, Zhong-Zhi Li, Yingying Zhang, Le Song, and Qianli Ma. LifelongAgentBench: Evaluating LLM agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025. URL https://arxiv.org/abs/2505.11942.

A SUPPORT DIAGNOSIS AND IDENTIFICATION

Coverage census protocol. Banks, tasks, ranking, and relevance rules are frozen without management. Relevant LAB pairs share the top-level SQL operation; CICIoT pairs share the true traffic class; these labels are used only for offline diagnosis. LAB's denominator is 7,686 unique pairs; CICIoT's 37,500 pairs are bank-specific across three seeds on one 1,000-task pool. Selected is Top-1 for LAB and Top-3 for CICIoT. Feedback additionally requires a parseable outcome, i.e., an output the checker can score; 283 of the 284 selected LAB pairs and 8,655/8,652 of the 8,655 selected CICIoT pairs (Qwen3.5-Flash/GPT-5.4-nano) were parseable, so Table 1 reports Selected. The controlled support simulation (Table A1) pairs utilities, tasks, noise, sample sizes, and seeds across Full/Narrow/Zero support at $n = 10{,}000$ and involves no LLM calls; the forced-selection probe is specified in Appendix A.2.

Construction of the controlled support simulation. Tasks have region indicator $z \sim \text{Bernoulli}(0.5)$. Without the target, $Y^- \sim \text{Bernoulli}(0.5)$. With the target, Y^+ has success probability 0.4 for $z = 0$ in both worlds; for $z = 1$ it is 0.9 in W^+ and 0.1 in W^-. Thus the population-average target effects are +0.15 and -0.25. Tasks, random draws, sample sizes, and seeds are paired across worlds. The target is always eligible for $z = 0$; for $z = 1$ its eligibility probability is 1, 0.04, or 0 under full, narrow, or zero support. Conditional on eligibility, it is selected with probability 0.5, giving target-selection probabilities of 0.5, 0.02, or 0 on $z = 1$. Binary outcomes use shared uniform draws against the corresponding success probabilities. Each log contains $n = 10{,}000$ tasks. The doubly robust estimator (Dudík et al., 2011) estimates the population-average effect, and the retain/remove decision follows its sign. RMSE and decision error are averaged over 100 seeds. Under zero support, all 100 paired logs are identical despite the opposite population-average effects, illustrating Proposition 1; Table A1 also reports its finite-sample uncertainty-expanded interval.

A.1 CONDITIONAL NON-IDENTIFICATION ARGUMENT

Let U be a target region with positive probability under the evaluation distribution. Fix a logging policy and all observed variables, including any memory content available to the manager. Assume that (i) the target is never selected on U under that logging policy; (ii) unselected potential outcomes are not constrained by additional content-to-utility, smoothness, or cross-context structural assumptions; and (iii) changing those unobserved outcomes does not change the law of logged variables. Assume also that the admissible target effects include both signs while preserving that logged-data law. Construct W^+ and W^- with the same logged-data law, but choose the target's with-memory potential success probabilities on U so that the average contextual effects have opposite signs. If a complementary outcome is observed, hold it fixed in both worlds.

$$\mathcal{L}(H_T \mid W^+) = \mathcal{L}(H_T \mid W^-), \qquad \delta_i(W^+) > 0 > \delta_i(W^-). \tag{A1}$$

Every measurable estimator or decision rule based only on H_T has the same output distribution in both worlds. Let p be the probability that a randomized rule retains the target. Its error probabilities are $1 - p$ in W^+ and p in W^-, so their average is 1/2 and the larger is at least 1/2. The statement concerns the specified target and retention-or-deletion decision loss. With extra external information or structural restrictions the conclusion may change (Sachdeva et al., 2020; Khan et al., 2024).

The effect range and its identified set are different objects. For example, let b_U be the identified mean success probability on U without the target and with its prescribed replacement. With binary outcomes and an unconstrained with-target mean in [0,1], the identified effect interval is $[-b_U, 1 - b_U]$. This interval contains both positive and negative effects when $0 < b_U < 1$; at either boundary it does not contain both signs. In the controlled construction, the sharp population identified set is $[-0.30, 0.20]$, of width 0.5; the reported mean width 0.568 additionally includes finite-sample uncertainty from Bonferroni-adjusted Wilson bounds on the observed stratum–action means, using empirical stratum weights and $[0, 1]$ for the unsupported mean. More data under low positive support can reduce error; more observations that never select the target do not create the missing outcome.

Proof sketch of Proposition 1. An estimator receives the same distribution of inputs in both worlds, so it cannot identify two different effects. Under an equally weighted two-world decision problem, at least one world has error probability at least one half.

Table A1: Controlled support comparison at $n = 10{,}000$.

Support	DR RMSE	Decision error	Point identification
Full	0.009	0%	Yes
Narrow	0.016	0%	Yes; larger finite-sample error
Zero	0.207	50%	No under the construction

Paired synthetic simulations over 100 seeds; no model calls. Under zero support, the population-average effects are +0.15/−0.25 and all paired logs are identical. The finite-sample uncertainty-expanded interval has mean width 0.568; empirical coverage and abstention are both 100%.

A.2 Forced-selection controls and dependence

Both probes use Qwen3.5-Flash, Natural, seed0; targets have natural $\mathrm{rank} > 10$ and were selected before new intervention outcomes were observed. LAB includes all 37 eligible pairs spanning 18 memories and 31 tasks. Its target-only context [m] is compared with historical/fresh natural [a] and empty memory. There are 99 new episodes: 37 target, 31 empty, and 31 fresh natural. CICIoT uses a deterministic maximum matching of 92 distinct memories and tasks from 3,485 candidate pairs. The forced context [a,b,m] retains the first two memories; controls are [a,b,c] and [a,b], for 276 new answers.

Table A2: All forced-selection comparisons.

Comparison	Positive	Negative	Unchanged	Pairs
LAB: forced − historical natural (primary)	3	1	33	37
LAB: forced − fresh natural	3	2	32	37
LAB: forced − empty	1	1	35	37
LAB: fresh − historical natural	3	2	32	37
CICIoT: forced − fresh natural (primary)	6	4	82	92
CICIoT: forced − historical natural	7	6	79	92
CICIoT: forced − drop-third	9	8	75	92
CICIoT: fresh − historical natural	3	4	85	92

Positive means incorrect-to-correct; negative means correct-to-incorrect. Unchanged means correctness unchanged, not identical text. Rows use shared samples and must not be added. LAB fresh-natural control is reused across pairs; at 31 unique tasks it is 3/2/26.

The LAB candidate census uses top-level SQL operation relevance, whereas its forced-selection subset uses matching full skill tags. CICIoT matching uses source/target class only offline. All interventions and controls in these probes were parseable. The experiment's role is to show that some previously unselected candidates change task correctness under forced selection.

Corroboration. As an idealized example of why the evidence requirements of §4.3 demand agreement, suppose two directional outcomes are independent, their true effects share the same sign, and each reports the wrong sign with probability $q < 1/2$. Conditional on the two outcomes agreeing, their sign-error probability is $q^2/[q^2 + (1-q)^2]$, approximately 0.155 when $q = 0.3$. Distinct tasks and nearest-neighbor groups do not by themselves ensure independence or a common effect sign, so the formula is illustrative; the practical effect of the requirement is evaluated through the lifecycle ablations (§6.3).

A.3 Service-feedback coverage under Top-1 reuse

We prove Proposition 2 for arbitrary numbers of task classes, memories, and tasks. The result characterizes when repeatedly using a successful memory leaves other good entries without service feedback. The uncovered count depends on how many good memories each class contains and whether its task budget reaches the first good memory. No random ordering of memories is assumed.

Let the finite task set contain $K \geq 1$ disjoint classes. Class z has L_z distinct tasks and a disjoint memory bank of m_z entries, of which g_z are good, with $L_z \geq 1$ and $1 \leq g_z \leq m_z$. Every task is executed exactly once. Write $m_{z,j}$ for the entry at initial rank j within class z, and let local time t

count tasks from that class. The total numbers of tasks, memories, and good memories are defined below.

$$T = \sum_{z=1}^{K} L_z, \qquad N = \sum_{z=1}^{K} m_z, \qquad g_{\text{tot}} = \sum_{z=1}^{K} g_z.$$

All tasks within a class share the same strict semantic ranking; ties may instead use a fixed, task-independent rule. Retrieval is restricted to the corresponding class, either by class gating or because every same-class entry ranks above every cross-class entry. An exhausted class bank supplies no memory and does not trigger retrieval from another class. Deleting an entry preserves the relative order of survivors. All memories are initially active. There are no memory additions, exploration steps, or other changes to the ranking.

These semantic assumptions have a simple geometric realization. Give task t in class z scalar embedding $u_{z,t}$ and memory j embedding $v_{z,j}$ as below, and use similarity $s(u, v) = -|u - v|$. Tasks lie to the left of all same-class memories, so increasing j gives the fixed ranking. Every within-class distance is below 1.25, while every cross-class distance exceeds 2.75. This construction works for arbitrary positive L_z and m_z; quality can be assigned to any subset of the ranked positions.

$$u_{z,t} = 4z - \frac{1}{4} + \frac{t}{8(L_z + 1)}, \qquad v_{z,j} = 4z + \frac{j}{m_z + 1}.$$

Each task uses the highest-ranked available entry. A success preserves that entry, whereas a failure removes it before the next task. Each task receives one attempt, with no immediate retry. Reuse is local to the class, so tasks from different classes may be interleaved arbitrarily. Here, good and bad denote deterministic success and failure types: every good entry succeeds on every task in its class, every bad entry fails, and feedback is accurate and immediate. Appendix A.4 retains these retrieval and update rules while replacing deterministic outcomes with stochastic success.

Define R_z as the initial rank of the first good memory. At most $m_z - g_z$ bad memories can precede it, giving the following bound.

$$R_z = \min\{j : m_{z,j} \text{ is good}\}, \qquad 1 \le R_z \le m_z - g_z + 1.$$

For local tasks $t < R_z$, induction shows that the first $t - 1$ entries have been deleted and entry t is selected. This entry is bad, so the task fails and deletes it. If task R_z is reached, its selected entry is good and remains available after success. Every lower-ranked entry stays below it, and the good entry is therefore selected on every subsequent task of that class. Denoting the selected entry and outcome by $a_{z,t}$ and $y_{z,t}$, respectively, we obtain

$$a_{z,t} = m_{z,\min(t,R_z)}, \qquad y_{z,t} = 1\{t \ge R_z\}, \qquad 1 \le t \le L_z.$$

Thus class z selects exactly $\min(L_z, R_z)$ distinct entries and exactly one good entry if $L_z \ge R_z$; otherwise it selects no good entry. Let U_{cov} count good entries that are never selected across the complete task set. Summing their classwise counts proves Proposition 2:

$$U_{\text{cov}} = g_{\text{tot}} - \sum_{z=1}^{K} 1\{L_z \ge R_z\} = g_{\text{tot}} - K + \sum_{z=1}^{K} 1\{L_z < R_z\}.$$

The first term, $g_{\text{tot}} - K$, counts good entries left unused after one good entry is found in each class. The remaining sum counts classes whose first good entry has not yet been reached. Consequently, U_{cov} is positive exactly when some g_z exceeds one or some L_z is below R_z. It is zero exactly when every class has one good entry and every class reaches that entry. Arbitrary interleaving leaves the result unchanged because each task modifies only its own class's bank.

For fixed rankings, reaching every first good entry gives $U_{\text{cov}} = g_{\text{tot}} - K$. A budget that guarantees this equality for every possible arrangement of good and bad entries is

$$L_z \geq m_z - g_z + 1 \text{ for every } z.$$

This condition is also necessary for that uniform guarantee: placing all bad entries before all good entries attains $R_z = m_z - g_z + 1$. With equal class sizes m and g and sufficient budgets, the uncovered fraction among good memories is $1 - 1/g$.

The formula covers both uniform and nonuniform finite task compositions. Uniform composition has $L_z = T/K$ when T is divisible by K; nonuniform composition allows any positive integer counts summing to T. For example, take $K = 10$, $m_z = 10$, $g_z = 5$, and $T = 1{,}000$. Compare 100 tasks per class with 50 tasks in each of nine classes and 550 tasks in the remaining class. Both compositions execute every task once and provide at least six tasks per class. Since $R_z \leq 6$, both leave exactly 40 of 50 good memories unselected, regardless of their fixed rankings. Their unequal frequencies alone do not change this count.

Coverage and task performance measure different properties. Let S_z and F_z denote successful and failed tasks, and SR the overall success rate. The trajectory above gives

$$S_z = \max(0, L_z - R_z + 1), \qquad F_z = \min(L_z, R_z - 1).$$

$$\text{SR} = \frac{\sum_{z=1}^{K} S_z}{T}.$$

Once all classes reach their first good entries, failures total the fixed number $\sum_z (R_z - 1)$. Further tasks succeed, so SR approaches one as T grows while U_{cov} remains $g_{\text{tot}} - K$. Unselected good entries remain active: U_{cov} is a coverage count, distinct from the permanent-deletion rate D_G and from the task–memory-pair census in Table 1. Same-class good entries are interchangeable in this construction, so lack of selection alone does not imply lost task success or a strictly positive effect relative to replacing one good entry with another.

An IID extension replaces the fixed task composition with independent class draws having probabilities $\pi_z \geq 0$ summing to one. With fixed rankings, the number $L_z(T)$ of visits to class z is $\text{Binomial}(T, \pi_z)$, and may be zero. The coverage identity also holds for zero visits. Taking its expectation yields

$$\mathbb{E}[U_{\text{cov}}(T)] = g_{\text{tot}} - K + \sum_{z=1}^{K} \Pr\{\text{Binomial}(T, \pi_z) < R_z\}.$$

Uniform sampling sets $\pi_z = 1/K$; nonuniform sampling uses unequal probabilities. For a fixed finite bank, every class with positive probability eventually reaches its first good entry almost surely. If K_+ classes have positive probability, $U_{\text{cov}}(T)$ therefore converges almost surely to $g_{\text{tot}} - K_+$. In particular, positive probability for every class gives the same residual count $g_{\text{tot}} - K$.

A.4 Outcome-based attribution and deletion after the first failure

This construction separates difficulty confounding from the deletion of beneficial memories after stochastic failures. We study a fixed bank under Top-1 retrieval and a first-failure deletion rule. A G memory is conditionally better than a specified B replacement on the same task class. This model-defined advantage is distinct from the source-correct/source-incorrect labels used as proxies in the experiments; replacing a G memory with an equally effective G memory need not reduce success.

A.4.1 Model and heterogeneous classes

Let $K \geq 1$ be any finite number of task classes. Class z has g_z G memories, b_z B memories, m_z total memories, and L_z distinct service tasks, each executed once. The counts $g_z \geq 1$, $b_z \geq 0$, and $L_z \geq 0$ are integers; counts and difficulty may differ across classes. Classes with no G entries contribute zero and may be omitted; all retained classes have at least one G entry. Write g_{tot} for the total number of G memories and D_{good} for their deletion count. Within each class, every memory of

a given type has the same success probability. The assumptions below imply strictly positive failure probabilities for both types.

$$m_z = g_z + b_z, \qquad g_{\text{tot}} = \sum_{z=1}^{K} g_z.$$

$$0 \le p_{B,z} < p_{G,z} < 1, \qquad q_{a,z} = 1 - p_{a,z}, \qquad a \in \{G, B\}.$$

Each class has a fixed ranking shared by its tasks. Retrieval selects its first undeleted entry; success retains that entry, whereas failure deletes it before the next task. There is no retry, memory insertion, or cross-class backfill. Once a class is exhausted, an external default handles its remaining tasks without further deletions. Conditional on class and type, outcomes are independent Bernoulli draws with stationary probabilities. Feedback records execution success accurately; randomness lies in the outcomes. Expectations do not require cross-class independence; joint-event statements below use independent classes.

A.4.2 Difficulty and outcome-based attribution

For a target memory, write its contextual success probability as $b(x) + \tau_i(x)$, where $b(x)$ refers to the prescribed alternative context. With E_i denoting selection, the observed mean decomposes into the baseline solvability of selected tasks and the target effect on those tasks:

$$\mu_i^{\text{obs}} = \mathbb{E}[b(X) | \, E_i = 1] + \mathbb{E}[\tau_i(X) | \, E_i = 1].$$

For example, set G/B success probabilities to 0.50/0.20 on hard tasks and 0.80/0.50 on easy tasks. G has a same-condition advantage of 0.30, yet hard-task G and easy-task B produce identical Bernoulli outcomes. If G and B encounter hard tasks with probabilities α and β, their marginal success difference is $0.30 - 0.30(\alpha - \beta)$. These parameters allow the advantage to disappear, but not reverse. More observations estimate these mixtures more precisely; separating baseline and effect requires a conditional contrast or additional identifying assumptions. The deletion mechanism below operates even within a single difficulty class.

For arbitrary K, suppose both types have positive selection probability, and let Z denote the selected task's class. Let $\alpha_z = \Pr(Z = z \mid G \text{ selected})$ and $\beta_z = \Pr(Z = z \mid B \text{ selected})$ be their class weights, each nonnegative and summing to one. Their population conditional means μ_G and μ_B obey the following exact decomposition. The first term averages same-class advantages; the second reflects unequal selection to baseline difficulty. Neither term depends on there being exactly two difficulty levels or ten classes.

$$\mu_G - \mu_B = \sum_{z=1}^{K} \alpha_z (p_{G,z} - p_{B,z}) + \sum_{z=1}^{K} (\alpha_z - \beta_z) p_{B,z}.$$

A.4.3 Single-memory risk

Let $T_{z,i}^G$ count uses of the ith G memory up to and including its first failure. After it begins continuous service, surviving h available opportunities requires h consecutive successes. Its deletion probability is therefore

$$\Pr(T_{z,i}^G \le h) = 1 - p_{G,z}^h = 1 - (1 - q_{G,z})^h.$$

This expression is conditional on reaching the entry and having those opportunities. To count deletions across the bank, earlier entries' service times must also be included. Let $T_{z,i}^G$ and $T_{z,j}^B$ denote the numbers of uses up to and including the first failure of the ith G entry and the jth B entry in class z, respectively. Independently within each class,

$$T_{z,i}^G \sim \text{Geom}(q_{G,z}), \qquad T_{z,j}^B \sim \text{Geom}(q_{B,z}).$$

$$\Pr(W = n) = (1 - q)^{n-1} q, \qquad n \geq 1.$$

A.4.4 Optimal ordering and the finite-horizon bound

Proposition. For every finite budget vector $L = (L_1, \ldots, L_K)$, placing all B entries before all G entries in each class minimizes expected G deletion over fixed rankings. Proof. Within-type success probabilities are identical, so within-type permutations have the same distribution. Couple rankings by assigning each entry the same outcome sequence, preserving relative order within each type. Let $\mathcal{B}_{z,r}(\sigma)$ be the B entries preceding the rth G entry in ranking σ. That G entry is deleted at class-local time

$$\tau_{z,r}(\sigma) = \sum_{i=1}^{r} T_{z,i}^{G} + \sum_{j \in \mathcal{B}_{z,r}(\sigma)} T_{z,j}^{B}.$$

Moving all B entries to the front adds their remaining positive waiting times to every such deletion time. Thus no G deletion occurs earlier, pathwise, at any deadline. Taking expectations proves optimality. The ordering delays beneficial selection; its objective is deletion count, not task success. Summing deletion indicators gives the attainable minimum below. Linearity of expectation permits summation across classes without assuming their independence.

$$\mathbb{E}[D_{\text{good}}(L; \sigma)] \geq \sum_{z=1}^{K} e_z(L_z).$$

$$e_z(L) = \sum_{r=1}^{g_z} \Pr\left(\sum_{j=1}^{b_z} T_{z,j}^{B} + \sum_{i=1}^{r} T_{z,i}^{G} \leq L\right).$$

A.4.5 Exact recursion

For the minimizing order, let $q_{z,j}$ equal $q_{B,z}$ for the first b_z positions and $q_{G,z}$ thereafter. Let $\pi_z(\ell, j)$ be the probability that j entries have been deleted after ℓ tasks. Initialize $\pi_z(0, 0) = 1$ and all other states to zero. For each step, use

$$\pi_z(\ell + 1, 0) = (1 - q_{z,1}) \pi_z(\ell, 0).$$

$$\pi_z(\ell + 1, j) = (1 - q_{z,j+1}) \pi_z(\ell, j) + q_{z,j} \pi_z(\ell, j - 1), \qquad 1 \leq j < m_z.$$

$$\pi_z(\ell + 1, m_z) = \pi_z(\ell, m_z) + q_{z,m_z} \pi_z(\ell, m_z - 1).$$

The exhausted state is absorbing. Deleting a prefix of length j removes $\max\{j - b_z, 0\}$ G entries, giving

$$e_z(L) = \sum_{j=0}^{m_z} \max\{j - b_z, 0\} \pi_z(L, j).$$

A.4.6 Finite and asymptotic consequences

For a class containing G entries, the minimum is zero when $L_z \leq b_z$ and strictly positive when $L_z \geq b_z + 1$: the earliest G deletion requires $b_z + 1$ failures, an event with positive probability. Thus task counts control the magnitude, and sufficiently short budgets give a zero bound. Under independent draws across classes, every finite budget also admits a positive-probability event on which all used G entries succeed. The deterministic minimum deletion count is therefore zero, despite a potentially large expectation.

With a fixed finite bank and indefinitely continuing opportunities in every class, all positive-parameter geometric waiting times are finite almost surely. Their finite sums are also finite, so every G entry is eventually deleted almost surely, and expected deletion converges to g_{tot}. Positive failure risk and continued selection are essential: an infallible entry can survive forever or block later entries. This limit extends the task stream; it does not assert complete deletion within an arbitrary finite dataset.

$$D_{\text{good}}(L;\sigma) \to^{\text{a.s.}} g_{\text{tot}}, \qquad \mathbb{E}[D_{\text{good}}(L;\sigma)] \to g_{\text{tot}}.$$

A.4.7 Numerical instance

Take ten classes, five hard and five easy, with five G and five B entries per class, using the success probabilities above. There are fifty G entries. The exact recursion gives minimum expected deletion counts 10.62833115, 34.11107054, 49.36883040, and 49.99983227 for equal per-class budgets $L = 10, 20, 50, 100$. At $L = 100$, a hard class contributes approximately five deletions and an easy class contributes 4.9999664542. These are analytic counts under the stipulated model, not empirical deletion rates.

Let $H_z(L) = \pi_z(L, m_z)$, the probability of exhausting class z under the minimizing order. Its final entry is G, so this also gives complete G deletion there. Coupling yields the following bound for any ranking when classes are independent; at $L = 100$ in the numerical instance its value is 0.9998609116, or 99.98609116%. The example demonstrates how repeated opportunities amplify a first-failure rule's stochastic deletion risk, while difficulty confounding remains a separate failure of attribution from raw outcomes.

$$\Pr\left(D_{\text{good}} = g_{\text{tot}}\right) \geq \prod_{z=1}^{K} H_z(L_z).$$

B Algorithm and implementation

Algorithm 1 specifies the CICIoT lifecycle. LEGAL tests audit eligibility, SCHEDULE allocates the shared budget, NEGATIVE-GATE checks deletion evidence, and LOW-FEEDBACK triggers review from centered service feedback. Table A3 summarizes the rules; Appendix B.1 gives implementation parameters and the LAB adaptation.

Algorithm 1 HABEAS control flow

1: Initialize the fixed bank, ACT states, behavioral histories, quarantine episodes, post-restoration review histories, and budgets.
2: For each service task: rank the full bank, filter by service state, backfill Top-k, execute once, and record the service outcome.
3: Update behavioral evidence. At a management checkpoint, quarantine never-restored entries satisfying the risk trigger; mark restored entries due for post-restoration review.
4: When a measured-pair slot is available, form audit-eligible targets using LEGAL and completed-task context alternatives under the current state.
5: Use SCHEDULE: semantic shortlist priority and, where applicable, prediction-distribution disagreement; share capacity with post-restoration reviews.
6: Execute or exactly reuse S^+ and S^-. Record positive, negative, tie, or invalid evidence without changing the service score.
7: For quarantined targets: apply the frozen NEGATIVE-GATE for deletion, or the two-task/two-nearest-neighbor-group positive evidence requirement with zero negative evidence for restoration.
8: For post-restoration reviews: a valid negative result returns the entry to quarantine; positive retains service; tie/invalid adds no support.
9: Log state transitions and consumed model calls/tokens; preserve unresolved quarantine episodes at the end of the run.

Table A3: Documented CICIoT rules and information permissions.

Item	Recorded rule
Risk check	Every 10 service tasks; never-restored entries require ≥ 5 uses and mean centered feedback ≤ -0.2.
Ordinary retrieval	Full-bank frozen ranking, active filtering, and backfill to at most three memories.
Restoration	Current quarantine episode: ≥ 2 positive task IDs, ≥ 2 nearest-neighbor group IDs, zero valid negative comparisons.
Deletion	Negative evidence on two tasks and two nearest-neighbor groups plus frozen deletion eligibility conditions, including raw $\mathrm{rank} \leq 10$.
Post-restoration review	10 new uses, or ≥ 5 with low shared task-level feedback; separate post-restoration review history.
Budget	≤ 1 measured pair per 10 tasks, ≤ 100 per 1,000 tasks, ≤ 80 semantic calls.
Quality information	Source quality and current truth are not supplied as an oracle to the selector. Completed-task checker results can score audits.
Priority signals	Semantic assessment and prediction-distribution disagreement prioritize eligible tests; they do not count as positive or negative audit evidence.
Evidence failure	Invalid/absent/tied evidence is not a negative audit outcome; pending is unavailable only when the state actually suspends retrieval.

B.1 Implementation parameters

For CICIoT, centered feedback is $u_t = r_t - b_{t-1}$, with $b_0 = 0.5$ and $b_t = 0.9b_{t-1} + 0.1r_t$, maintained separately for each method. Semantic ranking considers at most five quarantined candidates, comparing restored and filtered/backfilled Top-3 contexts from public features and untrusted historical predictions. It outputs a candidate-ID permutation; invalid output selects the queue head. Prediction distributions use normalized weights $w = 1/(0.05 + d)$, where d averages relative numeric-feature distances $|x - y|/\max(|x|, |y|)$ (zero when both values are zero) and an inequality indicator for Protocol Type. Contexts maximize total-variation disagreement, breaking ties by recency and task ID. Quarantine audits consider untested memory–task pairs from the last 50 tasks after the previous review, require a changed Top-3 context within the last 10 tasks, a 20-step cooldown, and three fresh eligible contexts after the first audit; each quarantine episode permits at most four pairs. The implementation does not reopen an exhausted episode; a new episode begins only when an active entry is quarantined again. Frontier priority is recent eligible-context count divided by one plus lifetime pair count. Ordinary slots alternate exploration and confirmation, with earliest first-positive evidence prioritized for confirmation; every fifth boundary gives priority to the oldest review. Deletion requires negative evidence on two tasks and two nearest-neighbor groups with target raw $\mathrm{rank} \leq 10$. Restored-entry reviews use a fresh actual-use context from the last 50 tasks after a 20-step cooldown. Every third ordinary slot, or an otherwise idle slot, serves due reviews in order of oldest review, greatest use count, then memory ID. A valid negative review re-quarantines the entry; other outcomes preserve its state. The triggers and shared budgets are given in Table A3.

Across the 24 main CICIoT runs (two models, four conditions, three seeds), 1,284 memory instances remain quarantined at the end of a run (Table A4). Classified in a fixed priority order, 52 exhausted the four-pair budget of their current episode, 473 had a valid negative pair in the episode without exhausting the budget (10 further entries had both), 15 had positive evidence from fewer than two tasks or nearest-neighbor groups, and the remaining 744 were eligible but did not reach a decision threshold within the run; every terminal entry met the eligibility rules at some checkpoint. No episode in these runs contained a valid negative pair followed by a valid positive pair. Quarantine therefore preserves content, but under the fixed protocol entries that exhaust an episode's budget remain unavailable.

LAB audits a memory by executing the same completed SQL task with and without the target under the Top-1 instantiation of the retrieval and backfill rules in §4.2. The paired executions hold the initial database state, task, model, and execution settings fixed. The LAB checker (LifelongAgentBench's DB Oracle) provides binary outcomes r^+ and r^-, yielding $\Delta = r^+ - r^-$: +1 means only the with-target execution succeeds, -1 means only the without-target execution suc-

Table A4: Terminal quarantine states in the main CICIoT runs.

Model	Regime	Terminal QUA (mean/run)	Budget exhausted	Negative in episode	Positive diversity <2	Eligible, undecided
Qwen3.5-Flash	N	142 (47.33)	11 (7.75%)	59 (41.55%)	2 (1.41%)	70 (49.30%)
Qwen3.5-Flash	S10	129 (43.00)	14 (10.85%)	58 (44.96%)	0 (0.00%)	57 (44.19%)
Qwen3.5-Flash	S20	148 (49.33)	6 (4.05%)	58 (39.19%)	3 (2.03%)	81 (54.73%)
Qwen3.5-Flash	S40	160 (53.33)	3 (1.88%)	68 (42.50%)	4 (2.50%)	85 (53.12%)
GPT-5.4-nano	N	174 (58.00)	3 (1.72%)	66 (37.93%)	1 (0.57%)	104 (59.77%)
GPT-5.4-nano	S10	178 (59.33)	3 (1.69%)	64 (35.96%)	2 (1.12%)	109 (61.24%)
GPT-5.4-nano	S20	176 (58.67)	6 (3.41%)	47 (26.70%)	2 (1.14%)	121 (68.75%)
GPT-5.4-nano	S40	177 (59.00)	6 (3.39%)	53 (29.94%)	1 (0.56%)	117 (66.10%)

Memory instances still quarantined at the end of a run, summed over three seeds (mean per run in parentheses). Categories are mutually exclusive in the listed priority order; percentages use the row total. No terminal entry was ineligible throughout its episode.

ceeds, and 0 means equal task outcomes and supplies no directional evidence. Audit executions are excluded from the service-success denominator. Quarantine uses the same trigger as on CICIoT: at checkpoints every ten tasks, a never-restored active entry with at least five uses and mean centered feedback ≤ -0.2 moves to QUA. Restoration is assessed using paired audit evidence, separately from service feedback. Deletion follows the CICIoT rule: negative evidence on two distinct tasks and two nearest-neighbor groups. Paired audit outcomes do not enter the service-feedback utility estimate. Audit-call counts include attempts and repairs. LAB's targeted allocation uses only the frontier priority and slot rotation and issues no semantic-ranking calls: LAB tasks are open-ended SQL generation judged by exact result or state matching, so there is no label space over which stored predictions could be compared, and the semantic-ranking and prediction-disagreement signals defined for CICIoT do not apply. HM applies the same check but deletes directly; it has no quarantine or restoration state.

On LAB, restoration requires, within the current quarantine episode, valid positive pairs from at least two distinct tasks and two distinct nearest-neighbor groups (on LAB, the nearest source memory under the fixed ranking) and no valid negative pair; a first positive result leaves the entry pending. A restored entry is retrievable everywhere and becomes due for review after ten uses since its last review, or after five uses with mean centered feedback ≤ -0.2, with at least twenty service steps between reviews. Review candidates are drawn from the last fifty completed tasks in which the target was retrieved after restoration and remains the current Top-1; the review compares the with-target Top-1 context with the without-target backfilled context. One valid negative review re-quarantines the entry, a positive review retains service, and ties or invalid pairs add no support. Reviews and quarantine audits share one budget of one pair per ten tasks, with every third ordinary slot reserved for due reviews and every fifth slot kept as a fairness slot.

In a recorded LAB audit, target memory src-114 was evaluated on task 341 at service step 110. The with-target context retrieved src-114, whereas excluding it backfilled the Top-1 position with src-105. Both sides were newly executed from the same task initialization. Although both trajectories issued the same SQL query, their submitted answers differed: the with-target side returned all ten requested rows, whereas the without-target side returned only five. The checker assigned outcomes of 1 and 0, yielding $\Delta = +1$. This provided positive evidence from one task and one nearest-neighbor group; the entry remained quarantined pending further confirmation. Each side used two model requests, with 1,829 and 2,137 tokens, respectively.

C Evaluation protocols and additional results

C.1 Benchmarks, scoring, and statistical units

CICIoT2023 supplies the traffic data (Neto et al., 2023); the evaluation is restricted to eight classes, 125 records per class, 46 numeric features, and a common pool of 1,000 tasks. Each of three seeds has 100 initial memories and varies source sampling and task order. The models are qwen3.5-flash-2026-02-23 and gpt-5.4-nano-2026-03-17.

The LAB-DB adaptation uses 300 tasks per run from LifelongAgentBench (Zheng et al., 2025). Natural banks contain 43 source-success entries and 7/6 source-failure entries for Qwen/GPT. SQL scoring ignores row order only when it is not requested and a complete answer is submitted within

three turns; column order, multiplicities, numeric requirements, explicitly ordered tasks, and failure boundaries remain enforced. HM uses the two-state history-based rule specified in Appendix B.1.

Rates are computed within each seed or block and then averaged with equal weight. Sample SD describes variability across three seeds, not a confidence interval. Resource use counts the model calls and input/output tokens required by each method, including audit attempts and repairs; an answer used by multiple methods is charged to each method. These counts are logical: cached or shared answers are still charged to the method that uses them, which is the sense of "logical" in the appendix tables. Audit tests and model calls are reported separately because a test can require multiple calls. Within each model and condition, differences are computed between runs matched by seed, bank, and task order, and we report the mean and sample SD of these paired differences over three seeds (Tables A8 and A19). The seeds share a task pool, and full-method parent runs reused in ablations are counted once. Five-block policy comparisons are summarized separately with equal block weights.

C.2 Metric definitions

Let G and B be disjoint initial source-quality sets, with controlled corruption assigned before natural quality so it is never double-counted. Let $Q_G(t)$, $D_G(t)$, and $A_B(t)$ be counts at time t; the initial set sizes remain the denominators. No-memory has no memory-state metrics. A clean bank with no incorrect entries has an undefined terminal incorrect-memory availability, not zero. E_B is the fraction of retrieved memories labeled incorrect, using all service retrieval slots as its denominator rather than the size of the remaining bank. Repeated selections of the same memory count separately; E_B is undefined if no memory is retrieved. In percentage tables, D_G and A_B abbreviate the corresponding rates, not the time-indexed counts.

$$D_G^{\text{rate}} = \frac{D_G(T)}{|G|}, \qquad U_G = \frac{Q_G(T) + D_G(T)}{|G|}, \qquad \overline{U}_G = \frac{\sum_{t=1}^{T} \left[Q_G(t^-) + D_G(t^-)\right]}{T\,|G|}. \tag{A2}$$

$$A_B^{\text{rate}} = \frac{A_B(T)}{|B|}, \qquad E_B = \frac{\sum_t |S_t \cap B|}{\sum_t |S_t|}. \tag{A3}$$

Time-averaged unavailability in the Natural ablation uses the service state before each selection. Event counts refer to state transitions, not unique memories. In Audit-Delete, clearing pending status is not a restoration because retrieval was never suspended. Means and sample SD are calculated from per-run rates.

Table A5: Natural CICIoT source-quality counts by seed.

Model	Seed	Initial G/B	HM D_G	HABEAS D_G/Q_G	HM/HABEAS active B
Qwen3.5-Flash	0	52/48	13	0/5	8/9
Qwen3.5-Flash	1	56/44	9	0/9	12/12
Qwen3.5-Flash	2	43/57	14	0/11	8/8
GPT-5.4-nano	0	36/64	6	0/4	13/11
GPT-5.4-nano	1	37/63	8	0/8	9/9
GPT-5.4-nano	2	43/57	10	0/8	9/9

Counts are not pooled to compute paper rates. The Oracle reference retains the G entries without replenishment.

Table A6: Success of the five compared methods on LAB-DB under the unified scoring rule.

Model	Regime	No-memory	RAG (No management)	HM (Baseline)	HABEAS (ours)	Oracle*
Qwen3.5-Flash	N	79.44	75.44	73.33	74.67	77.89
Qwen3.5-Flash	S10	78.67	73.11	71.22	71.56	—
Qwen3.5-Flash	S20	78.67	67.00	66.89	69.44	—
Qwen3.5-Flash	S40	78.67	61.00	61.33	63.67	—
GPT-5.4-nano	N	88.89	87.44	88.00	88.56	88.89
GPT-5.4-nano	S10	88.89	83.78	82.56	84.56	—
GPT-5.4-nano	S20	88.89	78.56	79.22	79.89	—
GPT-5.4-nano	S40	88.89	72.11	75.00	76.56	—

Success rate (%) on LAB-DB SQL tasks (300 tasks per run) under the unified scoring rule (all methods scored by the same SQL checker, Appendix C.1); equal-weight means over three seeds. Models as in Table 2. N = Natural; S10/S20/S40 = Stress10/20/40: injected erroneous principles target potential Top-1 selection on 10%/20%/40% of tasks (30/60/120 tasks here), using the shared protocol in Appendix C.3. Methods as in Table 2, with Top-1 retrieval. *Human-audited Oracle is reported only for Natural. Qwen3.5-Flash No-memory has a Natural reference of 79.44 and a shared Stress reference of 78.67; GPT-5.4-nano uses 88.89 across conditions.

Table A7: LAB success by seed; repeated references are labeled.

Model	Method	Regime	Mean %	Seed 0 / 1 / 2 (%)
Qwen3.5-Flash	No-memory	N	79.44	79.00 / 80.00 / 79.33
Qwen3.5-Flash	No-memory	S10	78.67	76.67 / 80.00 / 79.33
Qwen3.5-Flash	No-memory	S20	78.67	reused reference
Qwen3.5-Flash	No-memory	S40	78.67	reused reference
Qwen3.5-Flash	RAG (No management)	N	75.44	77.33 / 77.00 / 72.00
Qwen3.5-Flash	RAG (No management)	S10	73.11	73.00 / 75.00 / 71.33
Qwen3.5-Flash	RAG (No management)	S20	67.00	64.00 / 68.67 / 68.33
Qwen3.5-Flash	RAG (No management)	S40	61.00	60.67 / 61.33 / 61.00
Qwen3.5-Flash	HM (Baseline)	N	73.33	74.67 / 73.67 / 71.67
Qwen3.5-Flash	HM (Baseline)	S10	71.22	71.00 / 72.00 / 70.67
Qwen3.5-Flash	HM (Baseline)	S20	66.89	68.33 / 66.67 / 65.67
Qwen3.5-Flash	HM (Baseline)	S40	61.33	60.00 / 63.00 / 61.00
Qwen3.5-Flash	HABEAS (ours)	N	74.67	73.00 / 76.67 / 74.33
Qwen3.5-Flash	HABEAS (ours)	S10	71.56	70.33 / 70.33 / 74.00
Qwen3.5-Flash	HABEAS (ours)	S20	69.44	69.00 / 69.33 / 70.00
Qwen3.5-Flash	HABEAS (ours)	S40	63.67	61.33 / 66.33 / 63.33
Qwen3.5-Flash	Oracle	N	77.89	78.33 / 77.00 / 78.33
GPT-5.4-nano	No-memory	N	88.89	89.00 / 88.67 / 89.00
GPT-5.4-nano	No-memory	S10	88.89	reused reference
GPT-5.4-nano	No-memory	S20	88.89	reused reference
GPT-5.4-nano	No-memory	S40	88.89	reused reference
GPT-5.4-nano	RAG (No management)	N	87.44	87.67 / 87.33 / 87.33
GPT-5.4-nano	RAG (No management)	S10	83.78	82.67 / 83.33 / 85.33
GPT-5.4-nano	RAG (No management)	S20	78.56	77.00 / 80.67 / 78.00
GPT-5.4-nano	RAG (No management)	S40	72.11	72.67 / 73.00 / 70.67
GPT-5.4-nano	HM (Baseline)	N	88.00	88.33 / 87.67 / 88.00
GPT-5.4-nano	HM (Baseline)	S10	82.56	82.33 / 83.00 / 82.33
GPT-5.4-nano	HM (Baseline)	S20	79.22	81.00 / 79.00 / 77.67
GPT-5.4-nano	HM (Baseline)	S40	75.00	74.67 / 77.00 / 73.33
GPT-5.4-nano	HABEAS (ours)	N	88.56	87.67 / 90.00 / 88.00
GPT-5.4-nano	HABEAS (ours)	S10	84.56	84.67 / 85.00 / 84.00
GPT-5.4-nano	HABEAS (ours)	S20	79.89	80.00 / 79.33 / 80.33
GPT-5.4-nano	HABEAS (ours)	S40	76.56	76.67 / 77.67 / 75.33
GPT-5.4-nano	Oracle	N	88.89	89.33 / 89.67 / 87.67

Each seed has 300 tasks; Oracle is evaluated only under Natural. No-memory uses the same reference across Stress conditions.

Table A8: CICIoT success by seed: HABEAS versus HM.

Regime	HABEAS (%) Seed 0 / 1 / 2	HM (%) Seed 0 / 1 / 2	HABEAS − HM (pp) Seed 0 / 1 / 2	Difference (pp) Mean ± SD
Qwen3.5-Flash				
N	80.9 / 69.7 / 70.8	67.8 / 69.7 / 65.5	+13.1 / 0.0 / +5.3	+6.13 ± 6.59
S10	84.1 / 72.1 / 70.8	71.9 / 72.0 / 59.8	+12.2 / +0.1 / +11.0	+7.77 ± 6.67
S20	84.3 / 72.8 / 61.7	84.5 / 72.8 / 60.4	−0.2 / 0.0 / +1.3	+0.37 ± 0.81
S40	73.9 / 68.6 / 60.6	73.9 / 69.1 / 54.0	0.0 / −0.5 / +6.6	+2.03 ± 3.96
GPT-5.4-nano				
N	56.4 / 58.7 / 64.2	51.0 / 58.7 / 61.6	+5.4 / 0.0 / +2.6	+2.67 ± 2.70
S10	52.1 / 54.8 / 62.9	51.7 / 53.5 / 59.5	+0.4 / +1.3 / +3.4	+1.70 ± 1.54
S20	57.3 / 53.3 / 59.3	54.8 / 52.5 / 57.5	+2.5 / +0.8 / +1.8	+1.70 ± 0.85
S40	55.6 / 58.7 / 60.7	54.2 / 58.7 / 60.7	+1.4 / 0.0 / 0.0	+0.47 ± 0.81

Success (%) per seed, 1,000 service tasks each; runs are paired by seed, bank, and task order. Differences are HABEAS minus HM in percentage points (pp); mean ± sample SD over the three paired differences. Seeds share the task pool.

Table A9: LAB Qwen3.5-Flash terminal quality across all conditions.

Regime	Method	D_G %	U_G %	A_B %
N	RAG (No management)	0.00	0.00	100.00
N	HM (Baseline)	11.63	11.63	80.95
N	HABEAS (ours)	0.78	10.08	71.43
N	Oracle	0.00	0.00	—
S10	RAG (No management)	0.00	0.00	100.00
S10	HM (Baseline)	6.20	6.20	97.92
S10	HABEAS (ours)	0.78	9.30	93.06
S20	RAG (No management)	0.00	0.00	100.00
S20	HM (Baseline)	6.98	6.98	93.06
S20	HABEAS (ours)	0.00	5.43	85.42
S40	RAG (No management)	0.00	0.00	100.00
S40	HM (Baseline)	2.33	2.33	79.86
S40	HABEAS (ours)	0.00	4.65	80.56

All incorrect entries combine natural and controlled errors. No-memory: not applicable. Human-audited Oracle: no incorrect denominator.

Table A10: LAB GPT-5.4-nano terminal quality across all conditions.

Regime	Method	D_G %	U_G %	A_B %
N	RAG (No management)	0.00	0.00	100.00
N	HM (Baseline)	4.65	4.65	72.22
N	HABEAS (ours)	0.00	4.65	77.78
N	Oracle	0.00	0.00	—
S10	RAG (No management)	0.00	0.00	100.00
S10	HM (Baseline)	4.65	4.65	97.92
S10	HABEAS (ours)	0.00	2.33	92.36
S20	RAG (No management)	0.00	0.00	100.00
S20	HM (Baseline)	0.78	0.78	93.75
S20	HABEAS (ours)	0.00	3.10	83.33
S40	RAG (No management)	0.00	0.00	100.00
S40	HM (Baseline)	2.33	2.33	84.72
S40	HABEAS (ours)	0.78	2.33	78.47

All incorrect entries combine natural and controlled errors. No-memory: not applicable. Human-audited Oracle: no incorrect denominator.

C.3 STRESS INJECTION MECHANISM

Shared construction. CICIoT2023 and LAB-DB use the same selection-controlled injection mechanism. Natural retains the correct and incorrect memories produced by experience collection without artificial corruption. Stress places memories containing transferable erroneous principles into the initial bank, in the same experience format used for ordinary retrieval. Their content concerns traffic-classification rules in CICIoT and SQL reasoning rules in LAB. The injected memory supplies misleading advice to the agent; task inputs and the success checker remain those of the benchmark.

Selection calibration. For a fixed seed, we construct the controlled erroneous memories and configure their task coverage under the fixed retrieval ranking. Potential Top-1 selection is the fraction of service tasks whose first-ranked memory in the initial bank is a controlled erroneous memory, before any management changes the bank. Stress10, Stress20, and Stress40 set this target to 10%, 20%, and 40%, respectively. These targets correspond to 100/200/400 of the 1,000 CICIoT tasks and 30/60/120 of the 300 LAB tasks. The level is therefore defined over tasks: one injected memory can be the first-ranked candidate for multiple tasks, so the number of injected entries alone does not specify the Stress level.

Benchmark instantiation. The selection target is defined by the first-ranked candidate in both benchmarks. Service retrieval keeps each benchmark's retrieval configuration (§5.1): Top-3 for CICIoT and Top-1 for LAB. For CICIoT, the selection target does not specify the quality of the other two retrieved memories. The LAB Stress banks contain 43 source-success entries and 48 incorrect entries, combining natural and controlled errors, at every level. Their different Stress levels are determined by potential selection, rather than by interpreting 10/20/40 as the fraction of incorrect entries in the bank.

Shared conditions and realized selection. Each seed fixes the source sampling, controlled corruption, and task order; compared memory-management methods receive the same initial bank and retrieval ranking for that condition. The selection target is set before service begins. During service, quarantine, restoration, deletion, and backfill can change which memories are actually retrieved. We report their resulting task success and retrieved-incorrect share E_B separately. E_B counts both naturally incorrect and injected memories over actual retrieval slots, whereas the Stress level concerns controlled erroneous memories at the initial Top-1 position and uses tasks as its denominator. Thus potential selection specifies the initial challenge, while the final results measure how management responds to it.

C.4 MEMORY RETENTION AND SERVICE AVAILABILITY

Table A11: Terminal memory states in Natural.

Benchmark / model	Method	D_G (%) ↓	U_G (%) ↓	A_B (%) ↓
CICIoT / Qwen3.5-Flash	HM (Baseline)	24.54	24.54	19.32
CICIoT / Qwen3.5-Flash	HABEAS (ours)	0.00	17.09	20.02
CICIoT / GPT-5.4-nano	HM (Baseline)	20.51	20.51	16.80
CICIoT / GPT-5.4-nano	HABEAS (ours)	0.00	17.11	15.75
LAB / Qwen3.5-Flash	HM (Baseline)	11.63	11.63	80.95
LAB / Qwen3.5-Flash	HABEAS (ours)	0.78	10.08	71.43
LAB / GPT-5.4-nano	HM (Baseline)	4.65	4.65	72.22
LAB / GPT-5.4-nano	HABEAS (ours)	0.00	4.65	77.78

Terminal memory states (%) at the end of Natural runs, mean over three seeds; models as in Table 2. D_G: permanently deleted correct / initial correct; U_G: quarantined or deleted correct / initial correct; A_B: active incorrect / initial incorrect. Correct/incorrect are source labels (proxies for the good/bad memories of §1).

Table A11 compares the terminal memory states accompanying the task-performance differences. In CICIoT Natural, HM's correct-memory deletion rate D_G is 24.54%/20.51%; HABEAS's is zero, with 17.09%/17.11% held in quarantine where they remain recoverable (Table A11). Terminal incorrect-memory availability is similar: Qwen has 20.02% under HABEAS versus 19.32% under HM, GPT 15.75% versus 16.80%. On LAB Natural, Qwen HABEAS's D_G is 0.78% against HM's 11.63%, and GPT's terminal correct-memory unavailability (4.65%) equals HM's with no permanent deletion at all. These results show a lower correct-memory deletion rate under HABEAS;

quarantined entries retain their content and can be restored when the evidence gate is met within the available audit budget (Appendix B.1).

D AUDIT ALLOCATION AND COST

Allocation policy. On CICIoT, observable task features and memories' stated predictions define weighted label distributions p^+ and p^- for the two prospective contexts; the targeted policy favors larger total variation distance, $\mathrm{TV}(p^+, p^-) = \frac{1}{2}\sum_c |p^+(c) - p^-(c)|$, and semantic ranking prioritizes eligible candidates. These ranking signals prioritize tests; state decisions require audit outcomes. Restoration audits and post-restoration reviews share caps on measured pairs per service task and on semantic calls (Appendix B).

Answer reuse. A completed service answer supplies one side of a pair only when the task, the ordered retrieval context, and the execution settings match exactly; every tenth pair executes both sides afresh. Across the 24 main CICIoT runs, 1,731 of 2,269 pairs (76.29%) reused one side (265 the with-target side, 1,466 the without-target side) and 538 executed both sides afresh, 215 of them under the every-tenth rule. The reuse group yielded 150/430/1,150/1 positive/negative/tied/invalid pairs and the fresh group 48/110/380/0; the two groups are not randomly assigned, so these are descriptive counts. Reused answers are charged to the audit as logical calls and are not counted twice; each of the 2,807 newly issued audit requests completed in one attempt.

The audit-policy comparison asks what active evidence acquisition buys. On CICIoT, Uniform matches or exceeds B0 in all eight cells and Targeted matches or exceeds B0 in six, with the largest gain, +10.07 points, for Qwen3.5-Flash Stress10, where Targeted also improves over Uniform by 7.60 points. The two allocators differ in what they buy: Targeted attains the lowest terminal correct-memory unavailability in six of eight CICIoT cells (Qwen3.5-Flash Natural 17.09% against Uniform's 19.61%, at 73.80% versus 73.97% success), whereas Uniform spreads pairs evenly. On LAB, Targeted reaches comparable success with a fraction of Uniform's logical audit calls in every cell (6.33 versus 28.33 for Qwen3.5-Flash Natural); on LAB, Targeted uses only the frontier priority and slot rotation (Appendix B.1).

In CICIoT Natural, HABEAS consumes 179.00/177.33 extra model calls and 2.943/2.698 million total tokens per Qwen3.5-Flash/GPT-5.4-nano run. Uniform consumes 118.33/119.67 extra calls and 2.427/2.250 million tokens. Semantic ranking accounts for much of this difference: the targeted allocator spends roughly half a million additional tokens per run to choose where to audit. On LAB, audit tokens are a small fraction of the run: Natural audit-token totals are 2,518/3,662 against total tokens 986,019/696,759. All costs are reported as model calls and tokens per run.

The comparison covers both benchmarks, both models, and Natural and Stress10/20/40 (Appendix C.3). Targeted reports the full HABEAS results from the main comparison. Each result aggregates three seeds; success is reported as mean $\pm$ sample SD. Each LAB and CICIoT run contains 300 and 1,000 service tasks, respectively. Audit events and model calls are reported separately; event counts are taken from the experiment logs.

Table A12: All audit-policy success comparisons.

Benchmark	Model	Regime	B0	Uniform	Targeted
LAB	Qwen3.5-Flash	N	74.56 ± 2.34	75.33 ± 0.58	74.67 ± 1.86
LAB	Qwen3.5-Flash	S10	71.11 ± 1.35	71.56 ± 0.51	71.56 ± 2.12
LAB	Qwen3.5-Flash	S20	66.67 ± 0.67	68.67 ± 1.20	69.44 ± 0.51
LAB	Qwen3.5-Flash	S40	65.22 ± 2.83	61.67 ± 1.76	63.67 ± 2.52
LAB	GPT-5.4-nano	N	88.89 ± 1.26	87.89 ± 1.84	88.56 ± 1.26
LAB	GPT-5.4-nano	S10	83.11 ± 0.84	83.56 ± 0.51	84.56 ± 0.51
LAB	GPT-5.4-nano	S20	82.00 ± 0.33	81.00 ± 0.58	79.89 ± 0.51
LAB	GPT-5.4-nano	S40	77.89 ± 1.07	77.11 ± 0.51	76.56 ± 1.17
CICIoT	Qwen3.5-Flash	N	66.67 ± 7.55	73.97 ± 2.20	73.80 ± 6.17
CICIoT	Qwen3.5-Flash	S10	65.60 ± 7.03	68.07 ± 7.12	75.67 ± 7.33
CICIoT	Qwen3.5-Flash	S20	71.00 ± 3.52	73.23 ± 6.96	72.93 ± 11.30
CICIoT	Qwen3.5-Flash	S40	65.80 ± 10.04	66.47 ± 9.00	67.70 ± 6.70
CICIoT	GPT-5.4-nano	N	55.47 ± 9.22	55.63 ± 9.03	59.77 ± 4.01
CICIoT	GPT-5.4-nano	S10	56.63 ± 4.78	60.17 ± 2.73	56.60 ± 5.62
CICIoT	GPT-5.4-nano	S20	57.83 ± 2.65	59.40 ± 1.67	56.63 ± 3.06
CICIoT	GPT-5.4-nano	S40	57.00 ± 2.16	57.10 ± 2.04	58.33 ± 2.57

Percent, mean ± sample SD over three seeds. SD is descriptive.

Table A13: Audit-policy comparison: terminal memory states and retrieval selection.

Bench.	Model	Cond.	Method	D_G	Q_G	U_G	A_B	Q_B	D_B	E_B (%)
LAB	Qwen3.5-Flash	N	B0	0.00	10.08	10.08	76.19	23.81	0.00	11.78
LAB	Qwen3.5-Flash	N	Uniform	3.10	2.33	5.43	85.71	4.76	9.52	11.78
LAB	Qwen3.5-Flash	N	Targeted	0.78	9.30	10.08	71.43	23.81	4.76	10.44
LAB	Qwen3.5-Flash	S10	B0	0.00	8.53	8.53	91.67	8.33	0.00	20.11
LAB	Qwen3.5-Flash	S10	Uniform	0.78	3.10	3.88	92.36	5.56	2.08	20.67
LAB	Qwen3.5-Flash	S10	Targeted	0.78	8.53	9.30	93.06	6.25	0.69	21.11
LAB	Qwen3.5-Flash	S20	B0	0.00	6.98	6.98	86.11	13.89	0.00	28.11
LAB	Qwen3.5-Flash	S20	Uniform	0.00	2.33	2.33	88.19	8.33	3.47	30.00
LAB	Qwen3.5-Flash	S20	Targeted	0.00	5.43	5.43	85.42	9.72	4.86	28.67
LAB	Qwen3.5-Flash	S40	B0	0.00	4.65	4.65	79.17	20.83	0.00	38.33
LAB	Qwen3.5-Flash	S40	Uniform	0.00	0.78	0.78	84.72	5.56	9.72	44.33
LAB	Qwen3.5-Flash	S40	Targeted	0.00	4.65	4.65	80.56	7.64	11.81	42.22
LAB	GPT-5.4-nano	N	B0	0.00	3.10	3.10	88.89	11.11	0.00	16.89
LAB	GPT-5.4-nano	N	Uniform	0.78	2.33	3.10	100.00	0.00	0.00	18.33
LAB	GPT-5.4-nano	N	Targeted	0.00	4.65	4.65	77.78	22.22	0.00	15.44
LAB	GPT-5.4-nano	S10	B0	0.00	4.65	4.65	93.06	6.94	0.00	25.56
LAB	GPT-5.4-nano	S10	Uniform	1.55	0.00	1.55	93.75	2.78	3.47	25.78
LAB	GPT-5.4-nano	S10	Targeted	0.00	2.33	2.33	92.36	3.47	4.17	26.22
LAB	GPT-5.4-nano	S20	B0	0.00	3.10	3.10	84.72	15.28	0.00	31.67
LAB	GPT-5.4-nano	S20	Uniform	0.00	0.78	0.78	84.72	6.25	9.03	32.22
LAB	GPT-5.4-nano	S20	Targeted	0.00	3.10	3.10	83.33	7.64	9.03	32.89
LAB	GPT-5.4-nano	S40	B0	0.00	2.33	2.33	75.69	24.31	0.00	42.44
LAB	GPT-5.4-nano	S40	Uniform	0.00	0.78	0.78	81.94	9.03	9.03	45.33
LAB	GPT-5.4-nano	S40	Targeted	0.78	1.55	2.33	78.47	9.03	12.50	44.56
CICIoT	Qwen3.5-Flash	N	B0	0.00	22.90	22.90	18.33	81.67	0.00	27.10
CICIoT	Qwen3.5-Flash	N	Uniform	0.00	19.61	19.61	19.14	79.41	1.45	25.34
CICIoT	Qwen3.5-Flash	N	Targeted	0.00	17.09	17.09	20.02	77.88	2.10	24.67
CICIoT	Qwen3.5-Flash	S10	B0	0.00	24.03	24.03	19.57	80.43	0.00	24.61
CICIoT	Qwen3.5-Flash	S10	Uniform	0.00	22.18	22.18	21.29	78.05	0.65	29.91
CICIoT	Qwen3.5-Flash	S10	Targeted	0.00	14.44	14.44	29.65	65.93	4.42	32.00
CICIoT	Qwen3.5-Flash	S20	B0	0.00	23.15	23.15	28.95	71.05	0.00	36.94
CICIoT	Qwen3.5-Flash	S20	Uniform	0.00	19.97	19.97	30.11	67.57	2.32	37.59
CICIoT	Qwen3.5-Flash	S20	Targeted	0.00	16.03	16.03	27.47	70.80	1.73	32.79
CICIoT	Qwen3.5-Flash	S40	B0	0.00	14.03	14.03	28.82	71.18	0.00	45.27
CICIoT	Qwen3.5-Flash	S40	Uniform	0.00	14.03	14.03	28.82	70.21	0.97	44.88
CICIoT	Qwen3.5-Flash	S40	Targeted	0.00	12.71	12.71	28.83	69.22	1.95	44.21
CICIoT	GPT-5.4-nano	N	B0	0.00	21.59	21.59	19.39	80.61	0.00	38.68
CICIoT	GPT-5.4-nano	N	Uniform	0.00	20.67	20.67	18.86	81.14	0.00	38.30
CICIoT	GPT-5.4-nano	N	Targeted	0.00	17.11	17.11	15.75	83.72	0.53	34.48
CICIoT	GPT-5.4-nano	S10	B0	0.00	27.50	27.50	16.94	83.06	0.00	40.20
CICIoT	GPT-5.4-nano	S10	Uniform	0.00	22.92	22.92	18.58	80.87	0.55	38.34
CICIoT	GPT-5.4-nano	S10	Targeted	0.00	20.73	20.73	19.05	80.44	0.51	42.31
CICIoT	GPT-5.4-nano	S20	B0	0.00	18.56	18.56	26.58	73.42	0.00	47.22
CICIoT	GPT-5.4-nano	S20	Uniform	0.00	15.23	15.23	25.13	74.39	0.48	45.64
CICIoT	GPT-5.4-nano	S20	Targeted	0.00	23.66	23.66	24.13	75.39	0.48	47.23
CICIoT	GPT-5.4-nano	S40	B0	0.00	17.48	17.48	32.92	67.08	0.00	59.73
CICIoT	GPT-5.4-nano	S40	Uniform	0.00	15.96	15.96	32.92	67.08	0.00	59.64
CICIoT	GPT-5.4-nano	S40	Targeted	0.00	23.28	23.28	29.91	69.66	0.43	59.28

Values are mean percentages over three seeds. D_G, Q_G and U_G use initial correct memories as denominator; A_B, Q_B and D_B use initial incorrect memories; E_B uses service retrieval slots. N denotes Natural. Targeted reports the full HABEAS results.

Table A14: Audit-policy comparison: model calls and token use.

Cond.	Method	Main calls	Audit events	Audit calls	Rank calls	Total calls	Total tokens (k)	Mgmt tokens (k)
LAB / Qwen3.5-Flash								
N	B0	766.67	0.00	0.00	0.00	766.67	1018.12	0.00
N	Uniform	758.33	28.33	28.33	0.00	786.67	1009.19	10.30
N	Targeted	747.00	6.33	6.33	0.00	753.33	986.02	2.52
S10	B0	743.67	0.00	0.00	0.00	743.67	986.59	0.00
S10	Uniform	749.33	28.33	28.33	0.00	777.67	1009.82	10.33
S10	Targeted	736.33	7.33	7.33	0.00	743.67	979.41	2.91
S20	B0	739.67	0.00	0.00	0.00	739.67	988.44	0.00
S20	Uniform	736.33	28.00	28.00	0.00	764.33	992.89	10.48
S20	Targeted	736.00	9.33	9.33	0.00	745.33	990.80	3.77
S40	B0	719.33	0.00	0.00	0.00	719.33	970.98	0.00
S40	Uniform	715.33	18.33	18.33	0.00	733.67	972.28	7.18
S40	Targeted	714.00	11.33	11.33	0.00	725.33	963.74	4.46
LAB / GPT-5.4-nano								
N	B0	652.67	0.00	0.00	0.00	652.67	696.81	0.00
N	Uniform	650.00	26.67	36.33	0.00	686.33	714.53	17.57
N	Targeted	650.67	3.33	6.33	0.00	657.00	696.76	3.66
S10	B0	649.67	0.00	0.00	0.00	649.67	700.69	0.00
S10	Uniform	653.67	22.33	33.33	0.00	687.00	720.43	16.10
S10	Targeted	649.33	4.67	7.00	0.00	656.33	707.88	3.58
S20	B0	651.00	0.00	0.00	0.00	651.00	703.43	0.00
S20	Uniform	657.33	13.33	15.33	0.00	672.67	717.37	6.42
S20	Targeted	655.67	9.33	10.67	0.00	666.33	722.38	4.80
S40	B0	655.00	0.00	0.00	0.00	655.00	719.74	0.00
S40	Uniform	659.00	21.00	25.33	0.00	684.33	739.16	11.45
S40	Targeted	654.33	11.33	15.00	0.00	669.33	730.03	6.94
CICIoT / Qwen3.5-Flash								
N	B0	1,000.00	0.00	0.00	0.00	1,000.00	2158.62	0.00
N	Uniform	1,000.00	95.00	118.33	0.00	1,118.33	2427.38	263.42
N	Targeted	1,000.00	95.67	118.00	61.00	1,179.00	2943.35	773.79
S10	B0	1,000.00	0.00	0.00	0.00	1,000.00	2151.66	0.00
S10	Uniform	1,000.00	95.00	123.33	0.00	1,123.33	2430.67	273.58
S10	Targeted	1,000.00	93.33	113.67	55.33	1,169.00	2877.25	710.38
S20	B0	1,000.00	0.00	0.00	0.00	1,000.00	2158.79	0.00
S20	Uniform	1,000.00	95.33	122.33	0.00	1,122.33	2435.99	273.41
S20	Targeted	1,000.00	92.67	114.33	63.33	1,177.67	2958.47	795.95
S40	B0	1,000.00	0.00	0.00	0.00	1,000.00	2138.73	0.00
S40	Uniform	1,000.00	96.00	121.33	0.00	1,121.33	2407.55	266.63
S40	Targeted	1,000.00	95.33	116.33	66.00	1,182.33	2951.98	808.79
CICIoT / GPT-5.4-nano								
N	B0	1,000.00	0.00	0.00	0.00	1,000.00	2008.88	0.00
N	Uniform	1,000.00	95.00	119.67	0.00	1,119.67	2249.69	241.15
N	Targeted	1,000.00	94.67	116.67	60.67	1,177.33	2697.72	698.74
S10	B0	1,000.00	0.00	0.00	0.00	1,000.00	2008.14	0.00
S10	Uniform	1,000.00	95.00	119.33	0.00	1,119.33	2249.26	241.05
S10	Targeted	1,000.00	95.00	117.33	64.67	1,182.00	2754.87	744.45
S20	B0	1,000.00	0.00	0.00	0.00	1,000.00	2017.24	0.00
S20	Uniform	1,000.00	95.00	118.67	0.00	1,118.67	2255.69	239.65
S20	Targeted	1,000.00	95.00	119.67	59.00	1,178.67	2729.93	717.39
S40	B0	1,000.00	0.00	0.00	0.00	1,000.00	2020.52	0.00
S40	Uniform	1,000.00	95.00	121.00	0.00	1,121.00	2266.06	245.41
S40	Targeted	1,000.00	94.67	119.67	64.33	1,184.00	2775.05	754.48

Per-run means over three seeds; k = 1,000 tokens. Each LAB/CICIoT run contains 300/1,000 service tasks. Audit events are counted from the run logs separately from model calls; a two-sided audit does not imply exactly two new calls when responses are reused or repairs are required. Main calls include repairs; management tokens comprise audit and semantic tokens. Logical consumption includes shared/cached responses; reused main answers are not counted twice. N denotes Natural; Targeted reports HABEAS. On LAB, audit events are paired SQL executions of the same task with and without the target (Appendix B.1).

E LIFECYCLE ABLATION DETAILS

Ablations use Qwen3.5-Flash on CICIoT Natural with a shared 1,000-task pool, three seeds, and 100 initial memories per run: temperature=0, thinking disabled, `max_tokens`=512, JSON output. w/o quarantine keeps pending entries retrievable. Single-positive restoration requires one valid positive and no valid negative within the current quarantine episode. w/o post-restoration management disables review tests and uses the original shared-task-feedback quarantine rule; allocated review opportunities remain idle. Full-method values are those in the main comparison.

Table A15: Natural lifecycle ablations: Qwen3.5-Flash on CICIoT2023.

Variant	Success ↑	Final U_G ↓	Mean U_G ↓	E_B (%) ↓	Extra model calls ↓
HABEAS (ours)	73.80 ± 6.17	17.09	11.31	24.67	179.00
w/o quarantine	57.57 ± 6.52	0.00	0.00	46.70	159.00
Single-positive restoration	72.23 ± 2.86	14.35	11.07	26.11	176.00
w/o post-restoration management	69.73 ± 1.05	20.29	14.32	27.46	181.00

CICIoT2023 Natural with Qwen3.5-Flash. Rates in %. Success: mean $\pm$ descriptive SD over three seeds. Extra model calls per 1,000 tasks. All four variants have $D_G = 0$. The three ablations remove quarantine, relax restoration to one positive pair, and remove post-restoration management, respectively. Removing post-restoration management disables both review tests and their dedicated feedback handling.

Table A16: Ablation: task performance and memory quality.

Metric	HABEAS (ours)	w/o quarantine	Single-positive restoration	w/o post-restoration management
Success (%)	73.80	57.57	72.23	69.73
Macro-F1 (%)	72.34	56.52	70.91	67.62
Permanent correct-memory deletion (%)	0.00	0.00	0.00	0.00
Terminal correct-memory unavailability (%)	17.09	0.00	14.35	20.29
Time-averaged correct-memory unavailability (%)	11.31	0.00	11.07	14.32
Terminal incorrect-memory availability (%)	20.02	100.00	24.88	21.41
Time-averaged incorrect-memory availability (%)	34.68	100.00	37.08	35.07
E_B (%)	24.67	46.70	26.11	27.46
Tasks retrieving ≥ 1 incorrect memory (%)	46.37	63.90	47.27	48.53

Per-seed rates are averaged equally over three seeds. Correct/incorrect memories are defined by source-quality labels.

Table A17: Ablation: lifecycle events, audit outcomes, and resource use.

Metric	HABEAS (ours)	w/o quarantine	Single-positive	w/o post-rest. mgmt.
Correct-memory restoration events	1.33	0.00	2.67	2.33
Incorrect-memory restoration events	0.00	0.00	1.00	0.33
Pending-state clearances for correct memories	0.00	0.67	0.00	0.00
Post-restoration correct-memory re-quarantines	0.00	0.00	0.00	1.67
Post-restoration incorrect-memory re-quarantines	0.00	0.00	0.00	0.33
Incorrect-memory selections while pending (slots)	0.00	901.67	0.00	0.00
Post-restoration incorrect-memory selections (slots)	0.00	0.00	63.00	1.67
Unresolved end-of-run entries	47.33	41.00	43.00	48.67
Measured pairs	95.67	91.33	95.00	92.67
Post-restoration review pairs	9.33	1.33	12.67	0.00
Logical semantic calls	61.00	59.00	61.00	66.67
Extra logical management calls	179.00	159.00	176.00	181.00
Logical service + management calls	1179.00	1159.00	1176.00	1181.00
Extra/main logical calls (%)	17.90	15.90	17.60	18.10
Total tokens (k)	2943.345	2883.168	2949.498	2973.925
Management tokens (k)	773.794	691.025	772.238	811.064
Positive pairs (%)	11.78	6.15	8.35	9.38
Negative pairs (%)	22.62	7.57	19.65	15.44
Tied pairs (%)	65.60	86.28	72.00	74.82
Invalid pairs (%)	0.00	0.00	0.00	0.36
Distinct tested memories	48.67	41.33	48.00	49.33
Idle review slots	0.00	0.00	0.00	2.67

Single-positive = Single-positive restoration; w/o post-rest. mgmt. = w/o post-restoration management. Means over three seeds, each with 1,000 service tasks. Lifecycle counts are events or retrieval slots. Positive/negative/tied/invalid percentages are per-seed averages over measured pairs. Logical consumption includes shared/cached responses; k = 1,000 tokens.

Table A18: Ablation: variability across seeds.

Metric	HABEAS (ours)	w/o quarantine	Single-positive restoration	w/o post-restoration management
Success (%)	73.80 ± 6.17	57.57 ± 6.52	72.23 ± 2.86	69.73 ± 1.05
Time-averaged correct-memory unavailability (%)	11.31 ± 4.60	0.00 ± 0.00	11.07 ± 3.61	14.32 ± 4.18
E_B (%)	24.67 ± 5.70	46.70 ± 4.62	26.11 ± 7.66	27.46 ± 7.91
Extra logical management calls	179.00 ± 13.75	159.00 ± 15.72	176.00 ± 8.54	181.00 ± 6.08

Mean ± sample SD over three seeds sharing the same 1,000-task pool. HABEAS values are those reported in the main comparison.

Table A19: Ablation: per-seed success and paired differences.

(a) Success (%) Variant	Seed 0	Seed 1	Seed 2	Mean ± SD
HABEAS (ours)	80.9	69.7	70.8	73.80 ± 6.17
w/o quarantine	62.9	59.5	50.3	57.57 ± 6.52
Single-positive restoration	75.5	70.2	71.0	72.23 ± 2.86
w/o post-restoration management	70.8	69.7	68.7	69.73 ± 1.05
(b) HABEAS − variant (pp) Variant	Seed 0	Seed 1	Seed 2	Mean ± SD
HABEAS (ours)	—	—	—	—
w/o quarantine	+18.0	+10.2	+20.5	+16.23 ± 5.37
Single-positive restoration	+5.4	−0.5	−0.2	+1.57 ± 3.32
w/o post-restoration management	+10.1	0.0	+2.1	+4.07 ± 5.33

Success (%) per seed on the shared 1,000-task pool. Differences are full method minus variant in percentage points (pp); each summary is the mean ± sample SD of its three seed values, with paired differences summarized in panel (b).

F MANAGEMENT-POLICY COMPARISON

The comparison evaluates Qwen3.5-Flash on five disjoint 1,000-task CICIoT blocks under Natural and Stress20 (Appendix C.3). Audit-Delete keeps flagged memories retrievable while awaiting evidence, whereas Simple-Reversible (Simple in tables) quarantines immediately. Both controls select their net-vote threshold from $\{1, 2\}$ using three Natural development blocks; HABEAS parameters remain unchanged. All methods share audit eligibility rules, interfaces, and per-block caps of 100 audit pairs and 80 semantic calls. Controls may redirect unused review opportunities to other eligible audits. Event counts are five-block totals; availability metrics average per-block rates with equal weight.

Table A20: Post-freeze, five-block management-policy comparison.

Regime	Method	Success ↑	Mean U_G ↓	D_G ↓	E_B (%) ↓	Tokens (M) ↓
N	Audit-Delete	52.18	0.00	0.00	51.17	3.024929
N	Simple	66.44	14.06	2.98	23.87	3.112678
N	HABEAS (ours)	67.32	14.49	0.00	23.93	2.978186
S20	Audit-Delete	52.48	0.42	0.56	59.93	3.005916
S20	Simple	65.24	12.71	6.11	36.33	3.118653
S20	HABEAS (ours)	68.20	11.67	0.00	35.79	2.979350

Qwen3.5-Flash on five 1,000-task CICIoT2023 blocks. N/S20 = Natural/Stress20 (Appendix C.3); Natural is the primary condition. Rates (%) are equal-weight block means. Mean U_G = time-averaged correct-memory unavailability; E_B = retrieved-incorrect share. Tokens = per-run consumption in millions. Simple = Simple-Reversible.

Table A21: Management-policy comparison: restorations, re-quarantines, and post-restoration selections.

Regime	Method	Restorations G/B	Re-quarantines G/B	Post-restoration selections G/B (slots)
N	Audit-Delete	—	—	—
N	Simple	14/6	6/5	346/125
N	HABEAS (ours)	7/2	0/1	374/67
S20	Audit-Delete	—	—	—
S20	Simple	13/20	7/14	274/235
S20	HABEAS (ours)	8/2	0/1	370/167

Totals over five blocks. G/B denote source-labeled correct/incorrect memories. Audit-Delete has no service-restoration mechanism (—); its pending clearances are 17/16 for Natural and 9/15 for Stress20. N denotes Natural.

Table A22: Management-policy comparison: measured-pair outcomes.

Regime	Method	Positive	Negative	Tie	Invalid	Total
N	Audit-Delete	33	50	374	15	472
N	Simple	20	102	360	1	483
N	HABEAS (ours)	48	72	362	1	483
S20	Audit-Delete	24	53	382	15	474
S20	Simple	33	87	362	2	484
S20	HABEAS (ours)	56	94	333	1	484

Totals over five blocks. Invalid outcomes are distinct from ties and negative outcomes. Audit outcomes are excluded from service-success denominators. N denotes Natural.

Table A23: Management-policy comparison: time-averaged correct-memory unavailability by block and overall.

Regime	Block	Initial G	Audit-Delete	Simple	HABEAS (ours)
N	0	49	0.00	18.84	19.14
N	1	42	0.00	4.79	6.95
N	2	47	0.00	17.81	15.64
N	3	46	0.00	16.48	17.50
N	4	46	0.00	12.37	13.22
N	Mean	—	0.00	14.06	14.49
S20	0	39	0.00	15.38	11.62
S20	1	34	0.00	6.50	7.35
S20	2	37	0.00	14.70	13.51
S20	3	35	0.00	16.00	16.46
S20	4	36	2.11	10.97	9.42
S20	Mean	—	0.42	12.71	11.67

Rates are percentages. Mean rows are equal-weight means of five unrounded per-block rates. Initial G is the initial number of source-labeled correct memories, shared by methods within each block and condition; — in mean rows denotes not applicable. N denotes Natural.

F.1 Continued reuse after restoration

Figure A1 follows four source-correct memories that HM permanently deletes and HABEAS restores. The figure reports their subsequent retrievals and the success difference between the two complete management policies on the corresponding post-restoration task subsets.

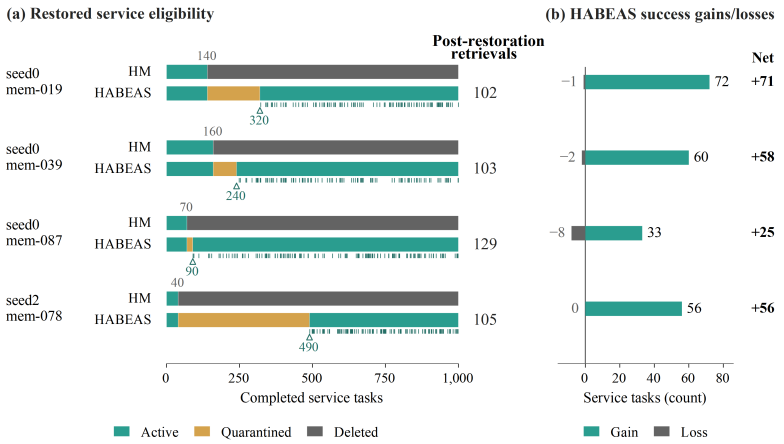


Figure A1: Continued reuse of restored memories and subsequent task gains. Four source-correct memories are permanently deleted by HM and restored by HABEAS under Qwen3.5-Flash on CICIoT2023 Natural. Triangles mark restoration; ticks mark subsequent service retrievals. Gain/Loss counts HABEAS-only/HM-only successes on these post-restoration tasks, paired by seed and task ID; Net is their difference. Retrievals exclude audits and reviews. Tasks can overlap across rows, so gains are not directly additive. These are policy-level differences on the selected task subsets, not isolated causal effects of individual memories.

G Evaluation under an alternative Stress mechanism

Skill-level corruption. We additionally evaluate GPT-5.4-nano on LAB-DB using seven persistent memories that prescribe misleading SQL procedures for insertion, update, deletion, aggregation, filtering, ordering, and projection. Examples include incrementing numeric values, negating predicates, and reversing sort directions. The source bank has 43 source-success and six source-failure entries; Stress adds the seven controlled records and matches them to task skills under a frozen ranking. Stress20 and Stress40 place a controlled record at the initial Top-1 position for 60 and 120 of the 300 service tasks, respectively, with nested selection sets.

Difference from the main mechanism. This construction gives explicit, skill-wide instructions presented as validated compatibility procedures. The main LAB Stress experiment instead constructs corrupted counterparts of source-success memories and appends transferable erroneous principles (Appendix C.3). Both mechanisms control potential Top-1 task selection; they differ in the content and number of injected records. All conditions here use seeds 0/1/2 with 300 tasks per seed and Top-1 retrieval. Success retains the benchmark's native scoring, including row-order-sensitive answer matching, whereas Table A6 uses the task-aware row-order normalization in Appendix C.1.

Management configuration. The supplementary LAB configuration, an early HABEAS variant labeled Early variant in Tables A24–A25, audits at most four memories every ten tasks, prioritizing uncertainty, proximity to decision thresholds, and staleness. Paired audit outcomes $\Delta = r^+ - r^-$ enter the utility estimate with weight 25; deletion requires $\Delta = -1$ and an updated upper bound at or below -0.2. HM uses the historical-mean deletion rule, while RAG retains the fixed bank without management. Method gains are compared within this protocol and its scoring rule.

Results. Under Stress20 and Stress40, the early variant achieves 80.11% and 79.22% success, exceeding HM by 3.22 and 1.33 percentage points and RAG by 12.33 and 24.55 points (Tables A24–A25). It deletes all seven controlled records while reducing correct-memory deletion from HM's 2.33%/3.10% to 1.55% in both conditions. HM retains one projection record with no potential se-

Table A24: GPT-5.4-nano on LAB-DB under Stress20 in the skill-corruption experiment.

Method	Success (%)	D_G (%)	A_C (%)	D_C (%)	Selections / 900
RAG	67.78 ± 1.17	0.00 ± 0.00	100.00 ± 0.00	0.00 ± 0.00	180
HM	76.89 ± 0.84	2.33 ± 2.33	14.29 ± 0.00	85.71 ± 0.00	93
Early variant	80.11 ± 0.51	1.55 ± 1.34	0.00 ± 0.00	100.00 ± 0.00	48

Mean ± sample SD over three seeds; 300 service tasks per seed. D_G is the permanently deleted fraction of the initial 43 source-correct memories. C is the seven injected records; A_C/D_C are their terminal active/deleted fractions, excluding naturally incorrect memories. Selections counts repeated retrievals of C across 900 task–seed observations.

Table A25: GPT-5.4-nano on LAB-DB under Stress40 in the skill-corruption experiment.

Method	Success (%)	D_G (%)	A_C (%)	D_C (%)	Selections / 900
RAG	54.67 ± 0.58	0.00 ± 0.00	100.00 ± 0.00	0.00 ± 0.00	360
HM	77.89 ± 1.71	3.10 ± 2.69	14.29 ± 0.00	85.71 ± 0.00	93
Early variant	79.22 ± 1.84	1.55 ± 1.34	0.00 ± 0.00	100.00 ± 0.00	68

Reporting conventions, metric definitions, and denominators are as in Table A24.

lection in this task pool, so its 14.29% record-level residual does not indicate continued selection here.

H DISCUSSION

Outcome-based attribution must distinguish task failure from a memory's contribution. HABEAS uses historical outcomes to trigger reversible quarantine and paired comparisons to support subsequent restoration and deletion decisions. The experiments evaluate this management choice jointly on reward, availability, realized selection, and cost.

Audit budget and cost. HABEAS treats evidence as a resource: all audits draw on one bounded budget, and the targeted allocator prioritizes entries for additional examination; on CICIoT, this includes contexts with greater predicted disagreement. The audit-policy comparison in Appendix D shows that active evidence acquisition pays—Uniform matches or exceeds the no-audit B0 in all eight CICIoT cells and Targeted in six, with gains of up to +10.07 points—and that on LAB the targeted allocator reaches comparable success with a fraction of a uniform allocator's audit calls. This evidence has a price that depends on how audits are instantiated. On CICIoT, an audit re-executes the task with and without the target memory and the budget is nearly fully used, so HABEAS consumes 34–38% more tokens than the no-audit configuration in every cell—two-thirds of it semantic ranking, with the audit-only Uniform configuration at 12–13% and, averaged over the eight CICIoT cells, 0.9 points below Targeted (Table A12). In the primary Natural condition this buys 6.13/2.67 points over history-based deletion and brings the correct-memory deletion rate from 24.54%/20.51% to zero (Tables 2 and A11). On LAB, audits are the same paired SQL executions but few (at most about eleven per run), so management accounts for 0.26–0.95% of total tokens and total consumption differs from the no-audit configuration by -3.15% to +2.69% (Table A14); in the Qwen3.5-Flash Natural cell the managed bank also needs fewer SQL turns (747 versus 767 service calls), so the audit cost is absorbed. Where a failed task costs further turns or retries, higher success offsets audit cost in this way, and the net token cost of management can be close to neutral, as it is on LAB; the complete accounting of model calls and tokens per run is in Appendix D.

What the audits reveal. Directional evidence is scarce: in the five-block runs 69–75% of measured pairs are ties (362 of 483 in Natural and 333 of 484 in Stress20; Table A22), and in the ablation runs 65.6% (Table A17), because a binary checker returns the same verdict on both sides whenever the task is solved or failed regardless of the target. This is why HABEAS acts only on corroborated pairs and why the budget matters. The directional minority is nevertheless decisive, and HABEAS extracts more of it than the reversible control from the same number of pairs (48 versus 20 positive pairs in Natural, 56 versus 33 in Stress20), consistent with targeted allocation placing audits where the two contexts are likely to disagree.

Fewer but better restorations. HABEAS restores fewer memories than Simple-Reversible (7 correct/2 incorrect versus 14/6 in Natural; 8/2 versus 13/20 in Stress20; Table A21), yet its restored memories are served more and are less often incorrect: 374 correct and 67 incorrect post-restoration selections versus 346 and 125 in Natural, 370 and 167 versus 274 and 235 in Stress20, with 0 correct-memory re-quarantines versus 6 and 7. Figure A1 shows the same effect at the level of individual memories that HM deletes and HABEAS keeps in service. The evidence gate therefore trades restoration count for restoration precision, and the ablations price the trade: Single-positive restoration raises terminal correct-memory availability (14.35% versus 17.09% unavailable) but admits one incorrect-memory restoration and 63 incorrect post-restoration selections per run and lowers success by 1.57 points; macro-F1 moves with success (72.34% versus 56.52%, 70.91%, and 67.62% for the three ablations; Table A16), so the gains are not an artifact of class imbalance.

What management can and cannot do. LAB-DB shows the boundary of the problem we solve. Injected errors make memory a liability there, and HABEAS recovers more of the damage than history-based deletion in every cell; but in some settings even the human-audited Oracle bank falls short of the No-memory reference (Qwen3.5-Flash on LAB-DB, 77.89% versus 79.44% in Natural; Table A6). Lifecycle management decides which memories to serve, not whether a bank is worth consulting: it makes memory safe to keep where memory helps, and limits the damage where it does not. Coupling it with an abstention decision is a natural next step (Appendix I).

I LIMITATIONS AND FUTURE WORK

Benchmark and model breadth. How much a memory manager helps depends on the task, and our evidence comes from two benchmarks (eight-class traffic classification and SQL tasks) and two fast

models, with three seeds per cell. The SQL benchmark also marks a boundary of the problem we solve: under Stress conditions no memory-using method reaches the No-memory reference, and for Qwen3.5-Flash even the human-audited Oracle bank (77.89%) stays below No-memory (79.44%) in Natural (Table A6); HABEAS decides which memories to serve, not whether to consult memory at all. Future work: evaluate on larger and more diverse benchmarks with continual memory writing, and combine lifecycle management with a decision on whether to consult memory at all.

Exploration of the Stress construction mechanism. The Natural condition evaluates management under naturally occurring memory errors. The main Stress conditions use designed erroneous-principle injection to control potential Top-1 selection; this construction does not reproduce how natural errors accumulate. Whether the gains persist as naturally occurring errors grow in share is therefore untested. Future work: collect banks in which natural errors reach higher shares through longer experience collection, and derive Stress gradients from observed memory-construction failures rather than from injected content.

Targeted allocation has no consistent advantage over uniform allocation. Targeted beats Uniform in some cells (Qwen3.5-Flash Stress10, +7.60 points) and trails it in others (GPT-5.4-nano Stress10/20, -3.57/-2.77; LAB GPT-5.4-nano Stress20, 79.89% versus 81.00%), and on CICIoT its semantic ranking adds roughly half a million tokens per run; its clearest benefit on CICIoT is availability rather than success (Appendix D). The priority signals—semantic relevance and prediction disagreement—are heuristics with no guarantee of finding the pairs whose outcome would change a decision. **Future work: further study of how audits should be allocated, including learning the allocation from audit outcomes**.

Availability cost and restoration errors. Quarantine holds 17.09%/17.11% of source-correct memories at the end of CICIoT Natural runs, time-averaged unavailability is 0.43 points higher than under Simple-Reversible in Natural, and two source-incorrect memories are restored per condition in the five-block runs (Tables A11, A20, A21). Reversibility preserves memory content and permits restoration when sufficient evidence is obtained; it does not guarantee eventual restoration, and under the fixed CICIoT protocol entries that exhaust an episode's audit budget remain unavailable (Table A4). Relaxing the gate recovers availability but admits incorrect-memory restorations (Single-positive restoration, §6.3). **Future work: restoration paths that recover availability sooner without admitting incorrect-memory restorations**.

Compute overhead. On CICIoT, audits add 179/177 model calls per 1,000 service tasks (about 18%) and raise token consumption by 34–38% over the no-audit configuration in every cell (Appendix D); two-thirds of the increase is the semantic ranking, which the allocation item above shows has no consistent advantage over uniform allocation, so the audit-only configuration costs 12–13%. On LAB the difference from the no-audit configuration is -3.15% to $+2.69\%$. Future work: reducing audit cost through reuse and adaptive budgets.