TOP-k RETRIEVAL LEAVES MEMORY VALUE UNDETERMINED

Anonymous authors
Paper under double-blind review

ABSTRACT

A memory-augmented agent must decide which memories to keep, merge and retrieve, and its only evidence of their value is its retrieval log: the memories it retrieved in each episode and whether it succeeded. Existing methods assign every memory a value from this log, even when the log cannot separate one memory's contribution from another's. We treat the log as an experiment that nobody designed and ask which memory values it determines. Two counts answer this before any outcome is observed: how often each memory was retrieved, and how often each pair was retrieved together. From these counts, we prove that top-k retrieval leaves every memory value undetermined, however many episodes are logged, and that a question with its own memories, asked once, determines at most one of their values. On fourteen standard memory benchmarks, no memory value is determined in any of 55 configurations. Both causes are choices about retrieval. We introduce LEDGER, which estimates each determined Shapley value in closed form with a confidence interval and reports every other value as undetermined. It merges memories with identical histories without losing a determined value, and it retrieves so that the next log determines more. On pooled HotpotQA, LEDGER raises the fraction of memories determined from 0.001 to 0.797, at a cost of 2.6 accuracy points on HotpotQA and with a gain of 2.7 on LoCoMo. Code: `https://anonymous.4open.science/r/ledger-review-AF71/`.

1 INTRODUCTION

Agents that operate over long horizons keep a store of memories and must decide which to retain, which to merge and which to retrieve for each query (Pink et al., 2025; Du et al., 2025). Each of these decisions depends on how much every memory contributes to the agent's success. A deployed agent can learn this only from its own history: in each episode it retrieved some memories and then succeeded or failed. This paper asks *what that history can tell us about the value of each memory, and when it can tell us nothing*.

Existing systems follow three approaches. The first scores memories with proxies such as recency, access frequency or similarity to the query (Wei et al., 2026a; Wu et al., 2026a;b). The second learns a value signal from downstream reward (Tan et al., 2025; Zhang et al., 2026; Yang et al., 2026). The third computes the Shapley value (Shapley, 1953), which divides the agent's success fairly among the memories it used (Wei et al., 2026b; Ye & Yoganarasimhan, 2025; Patel et al., 2025). Three challenges stand in the way. First, *a deployed agent cannot experiment on its memory*: Shapley methods need outcomes for sets of memories that were never retrieved together, and the agent cannot go back and retrieve them. Second, memories are retrieved in groups, so *the history may not separate the contribution of one memory from that of another*. When it does not, a method still returns a number, but the number is set by the method and not by the data, and nothing in the output reveals this. Third, *making the history more informative can lower accuracy*, because it means changing what the agent retrieves.

We treat the history as *an experiment that nobody designed*. Each episode is a trial in which a set of memories was retrieved and an outcome was observed, and the theory of experimental design states which quantities a set of trials determines (Rao, 1973; Pukelsheim, 2006). This view answers each challenge. First, it needs no new trials, because it asks only what the recorded ones determine. Second, it decides, for every memory, whether the history determines its value before any value is

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.

estimated, so a method can decline to report a value that the data do not support. Third, it identifies which changes to retrieval make the history informative, so their cost in accuracy can be measured.

The analysis needs only two counts that any retrieval log contains: how often each memory was retrieved, and how often each pair was retrieved together. From these counts alone, before any outcome is used, it decides which memory values the history determines. Applied to standard memory benchmarks (Yang et al., 2018; Maharana et al., 2024; Wu et al., 2025), it finds that none is determined. One common convention is enough to cause this: retrieving the same number of memories for every query. A second, giving each question its own memories, limits each question to as many determined values as it has episodes. Both conventions are choices about retrieval, so the remedy changes retrieval: share one store across queries, vary how many memories are retrieved, and fill part of each retrieval with the relevant memories retrieved least often. On pooled HotpotQA most values then become determined, fewer on other stores, while accuracy falls slightly on one corpus and rises on another. Our method, LEDGER, estimates the determined values with confidence intervals and reports the rest as undetermined.

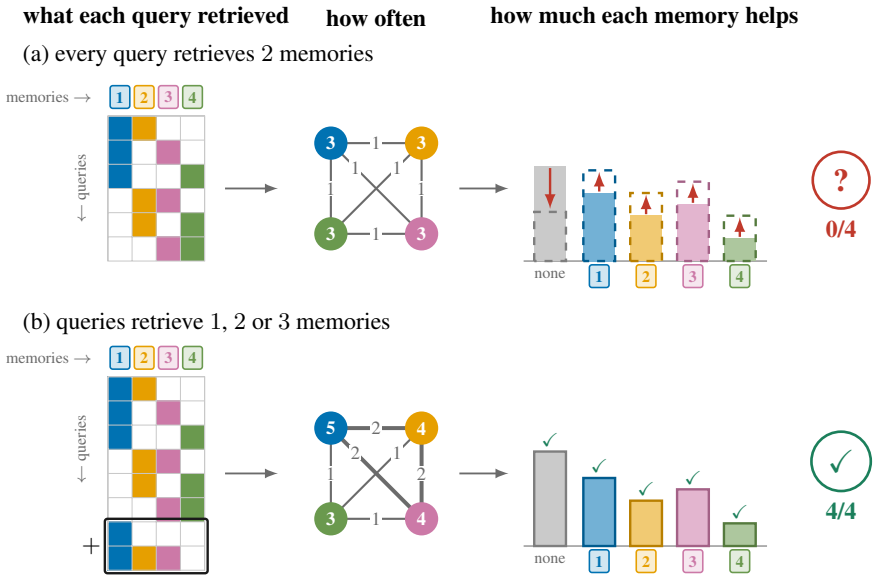

Figure 1: A history that looks sufficient and determines nothing. Rows are episodes, labelled by the memories they retrieve; the column $|S_t|$ gives each retrieval size. *Left:* each pair of four memories retrieved once. *Right:* the same episodes plus retrievals of sizes 1 and 3.

Figure 1 shows the first convention on four memories. Each pair is retrieved once, giving six episodes for four unknown values, which looks sufficient. Yet every episode retrieves two memories, so adding the same amount to every value and removing twice that amount from the baseline leaves every predicted outcome unchanged: the history fixes only the differences between values. One episode of size one and one of size three determine all four. Appendix H discusses the scope, applications and limitations of these results, and Appendix I collects the notation and terms.

This paper makes three contributions. First, it answers a question that prior work leaves open: which memory values a deployed agent's history determines. Proxy and learned methods report values without asking it, and Shapley methods avoid it by replaying sets of memories; we answer it exactly from the history, and show that a fixed retrieval size leaves every value undetermined however long the history grows (Section 3). Second, it gives LEDGER, which computes each determined value in closed form with a confidence interval, reports every other value as undetermined, and merges memories with identical histories without losing any determined value (Section 4). Third, it shows that the conventions the field uses to evaluate memory leave every memory value undetermined, and that changing how memories are retrieved makes most values determined on pooled HotpotQA and fewer on other stores, at a measured cost in accuracy (Sections 4 and 5). The same counts appear in slate off-policy evaluation, which sets the undetermined part aside because it does not affect the value of a policy (Swaminathan et al., 2017); for memory, the undetermined part is the finding.

2 Background and Problem Formulation

Preliminaries. We use three concepts from prior work. *Möbius dividends:* every set function v on the subsets of $[n] = \{1, \dots, n\}$ can be written uniquely as $v(S) = \sum_{B \subseteq S} \theta_B$, where the dividend θ_B is the part of the value that the set B creates jointly and none of its proper subsets creates (Harsanyi, 1963; Grabisch et al., 2000). *The Shapley value* of element i is the unique attribution satisfying efficiency, symmetry, null player and additivity (Shapley, 1953); in dividends it is $\varphi_i = \sum_{B \ni i} \theta_B / |B|$, so each dividend is shared equally among its members (Harsanyi, 1963). *Estimability:* in a linear model $\mathbf{r} = \mathbf{A}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, a combination $\mathbf{c}^\top \boldsymbol{\beta}$ is estimable when some linear function of $\mathbf{r}$ has expectation $\mathbf{c}^\top \boldsymbol{\beta}$ for every $\boldsymbol{\beta}$, which holds exactly when $\mathbf{c}$ lies in the row space of $\mathbf{A}$ (Rao, 1973, Section 4a.2); otherwise all least-squares solutions fit the data equally well and give $\mathbf{c}^\top \boldsymbol{\beta}$ every real value (Lemma A.2). Appendices A.1 and A.2 state these concepts formally.

Setting. An agent holds a store of n memories $m_1, \dots, m_n$, identified with $[n]$. In episode $t \in [T]$ it receives a query q_t drawn from a distribution $\mathcal{D}_Q$, retrieves a non-empty set $S_t \subseteq [n]$, answers, and observes an outcome $r_t \in [0, 1]$, for example whether the answer was correct. We make two assumptions; Appendix A.3 discusses what each requires and what is lost when it fails.

Assumption 1 (Outcome model)**.** For each query q there is a set function $v_q : 2^{[n]} \to [0, 1]$ such that $\mathbb{E}[r_t \mid \mathcal{F}_{t-1}, q_t, S_t] = v_{q_t}(S_t)$, where $\mathcal{F}_{t-1}$ is the history before episode t, and $\eta_t = r_t - v_{q_t}(S_t)$ is conditionally σ-sub Gaussian.

Assumption 2 (Resolution d)**.** The query-averaged value $v(S) = \mathbb{E}_{q \sim \mathcal{D}_Q}[v_q(S)]$ has $\theta_B = 0$ whenever $|B| > d$.

The baseline $\theta_\emptyset = v(\emptyset)$ is the expected outcome with nothing retrieved, and the value of memory i is its Shapley value φ_i under v, which the baseline does not enter.

Input and output. The method reads only what an agent already records: for each episode t, the retrieved set S_t and the outcome r_t. We store the sets as a binary matrix $\mathbf{X} \in \{0, 1\}^{T \times n}$, one row per episode and one column per memory, with $\mathbf{X}_{ti} = 1$ exactly when $m_i \in S_t$, and the outcomes as a vector $\mathbf{r}$. The product $\mathbf{X}^\top \mathbf{X}$ holds the two counts of the introduction,

$$\left(\mathbf{X}^\top \mathbf{X}\right)_{ii} = \mathrm{freq}(m_i), \qquad \left(\mathbf{X}^\top \mathbf{X}\right)_{ij} = \mathrm{co}(m_i, m_j), \tag{1}$$

how often m_i was retrieved and how often m_i and m_j were retrieved together. By Assumption 2, each outcome is the sum of the dividends of the retrieved sets of at most d memories plus noise $\varepsilon_t = r_t - v(S_t)$. This gives the linear model derived in Appendix A.4,

$$\mathbf{r} = \mathbf{X}_d \boldsymbol{\theta} + \boldsymbol{\varepsilon}, \qquad (\mathbf{X}_d)_{tB} = \mathbf{1}[B \subseteq S_t], \quad |B| \le d, \tag{2}$$

whose column for $B = \emptyset$ is all ones and carries the baseline; at $d = 1$, $\mathbf{X}_1 = [\mathbf{1}_T \mid \mathbf{X}]$. The output has one entry per memory: an estimate of φ_i with a confidence interval when the log determines it, and *undetermined* otherwise.

Problem. Given this log, we first ask which memory values it determines. Memory i is *identifiable* when the log determines φ_i: writing $\varphi_i = (\mathbf{c}^{(i)})^\top \boldsymbol{\theta}$ with $c^{(i)}_B = \mathbf{1}[i \in B]/|B|$, this holds exactly when $\mathbf{c}^{(i)}$ lies in the row space of $\mathbf{X}_d$ (Definition A.4), and it is decided from $\mathbf{X}_d$ alone, before any outcome is seen. The determined values then guide how the agent manages its store: which memories to merge without losing a value, which to keep under a budget (Appendix F.7), and how to retrieve so that the next log determines more, at a cost in accuracy that we measure.

3 What a Top-k Log Determines

The problem began with one decision for each memory: whether the log determines its value. This section makes that decision for the logs that agents actually produce. It first shows that whether $\mathbf{c}^{(i)}$ lies in the row space of $\mathbf{X}_d$ depends only on counts the agent already keeps, and then checks this condition under three common retrieval conventions: retrieving a fixed number of memories per query, giving each query its own memories, and retrieving some memories always together. For each convention, it identifies what the log cannot see and the change to retrieval that restores it. All proofs are in Appendix B.

What a log cannot see. By Section 2, memory i is identifiable exactly when $\mathbf{c}^{(i)}$ lies in the row space of $\mathbf{X}_d$, that is, when it is orthogonal to the kernel $\ker \mathbf{X}_d$. We call each $\mathbf{h} \in \ker \mathbf{X}_d$ an *invisible change*: $\boldsymbol{\theta}$ and $\boldsymbol{\theta} + s\mathbf{h}$ predict the same expected outcome in every episode for every $s \in \mathbb{R}$, while φ_i moves by $s\,(\mathbf{c}^{(i)})^\top \mathbf{h}$. Hence φ_i is determined exactly when $(\mathbf{c}^{(i)})^\top \mathbf{h} = 0$ for every $\mathbf{h} \in \ker \mathbf{X}_d$. Since $\ker \mathbf{X}_d = \ker(\mathbf{X}_d^\top \mathbf{X}_d)$ (Appendix B.1), the invisible changes depend only on $\mathbf{X}_d^\top \mathbf{X}_d$, which at $d = 1$ is

$$\mathbf{X}_1^\top \mathbf{X}_1 = \begin{bmatrix} T & \mathbf{f}^\top \\ \mathbf{f} & \mathbf{X}^\top \mathbf{X} \end{bmatrix}, \qquad f_i = \mathrm{freq}(m_i). \tag{3}$$

Whether each memory is identifiable is therefore decided by T, the retrieval frequencies and the co-retrieval counts alone, and at resolution d by the number of episodes that retrieve each set of at most $2d$ memories (Lemma B.1). The retrieval convention that produces these counts therefore decides which memory values a log can determine, before any outcome is observed and whatever the memories contain.

A fixed retrieval size. The most common convention retrieves the k memories most relevant to the query, so every episode retrieves the same number of memories; we call such a log a *top-k log*. In a top-k log, an episode that retrieves a set B of fewer than k memories retrieves exactly $k - |B|$ other memories with it. In every episode, therefore, the columns of $\mathbf{X}_d$ for the sets $B \cup \{j\}$ with $j \notin B$ sum to $k - |B|$ times the column for B, which gives the invisible change

$$\mathbf{h}_B \;=\; \sum_{j \notin B} \mathbf{e}_{B \cup \{j\}} \;-\; (k - |B|)\,\mathbf{e}_B \;\in\; \ker \mathbf{X}_d, \qquad |B| < d, \tag{4}$$

where $\mathbf{e}_B$ is the unit vector for θ_B. For $B = \emptyset$, $(\mathbf{c}^{(i)})^\top \mathbf{h}_\emptyset = 1$ for every memory i, so adding $\mathbf{h}_\emptyset$ changes every φ_i without changing any predicted outcome.

Theorem 1 (Top-k leaves every value undetermined)**.** *If every episode retrieves exactly k memories, no memory is identifiable, at any resolution d and any number of episodes T. If, in addition, outcomes are binary and each v_q takes values in $(0, 1)$, no estimator of any φ_i is consistent.*

At $d = 1$, the invisible change $\mathbf{h}_\emptyset$ raises every memory's value by s and lowers the baseline by ks; this is the ambiguity of Figure 1. No estimator is consistent because adding a small multiple of $\mathbf{h}_\emptyset$ to the dividends of every v_q leaves the distribution of the log unchanged, so two different values of φ_i produce the same data (Appendix B.2). More episodes do not help, because each new episode again retrieves exactly k memories. The invisible changes disappear only if the retrieval size varies, and the next proposition gives how many sizes are needed.

Proposition 1 (Sizes needed)**.** *If the retrieval sizes take at most d distinct values, then $\ker \mathbf{X}_d$ contains a nonzero invisible change at every T.*

At least $d + 1$ distinct sizes are therefore needed for the log to determine every dividend; at $d = 1$, two. With at most d sizes, all below $n/2$, no memory is identifiable at all (Corollary B.1). Under deterministic retrieval, where each query's sets are the top of one fixed ranking, the sets retrieved for a query at different sizes are nested, and nested sets can leave an invisible change of the same kind as $\mathbf{h}_\emptyset$, one that raises some memories' values and lowers the baseline (Lemma B.2). Section 4 removes both obstacles by varying the size and breaking the nesting, and Section 5.3 observes the nesting on real logs.

Private memories per query. The second convention gives each query b its own n_b memories. The T_b episodes of query b then retrieve only from these memories, so $\mathbf{X}$ splits into one block per query, and the block of query b has rank at most T_b.

Proposition 2 (Private memories)**.** *If each query b has its own n_b memories and T_b episodes, at least $\sum_b \max(n_b - T_b,\, 0)$ memories are not identifiable at $d = 1$.*

A query asked once determines at most one of its n_b memory values. Sharing one store across queries lifts this limit, but a fixed retrieval size leaves $\mathbf{h}_\emptyset$ in the log (Theorem 1), so a shared store must be combined with varied retrieval sizes.

Memories always retrieved together. The third convention follows from ranking by relevance: memories that rank high for exactly the same queries are retrieved in the same episodes, and their

columns of $\mathbf{X}$ are then identical. At $d = 1$, the invisible change $\mathbf{e}_{\{i\}} - \mathbf{e}_{\{j\}}$ raises φ_i and lowers φ_j by the same amount, leaving $\varphi_i + \varphi_j$ unchanged; this is aliasing (Rosenbaum & Zubizarreta, 2025). It accounts for the spurious credit that memory systems report for memories co-retrieved with useful ones (Yang et al., 2026; Yan et al., 2026a). Because this change leaves the sum unchanged, Section 4 merges such memories into one.

Retrieval decides what the log can determine. Theorem 1 and Propositions 1 and 2 identify, from counts alone, which memory values a log leaves undetermined and why. Each undetermined value comes from an invisible change, and each invisible change comes from a retrieval convention (Table 1): a fixed size gives $\mathbf{h}_B$, nested sets give a change like $\mathbf{h}_\emptyset$, private memories give changes inside each block, and tied memories give $\mathbf{e}_{\{i\}} - \mathbf{e}_{\{j\}}$. Every invisible change can be found from the log before any value is estimated, and a change to retrieval can remove it. Memory valuation thus becomes a question of design: which retrievals make the values an agent needs determinable.

Table 1: Retrieval conventions and their invisible changes. Each icon is a small log: rows are episodes, columns are memories, and a filled cell marks a retrieval.

Log	Convention	Invisible change	Undetermined	Remedy (Section 4)
	Fixed size k	$\mathbf{h}_B$, $\lvert B\rvert < d$	every value, at any T	$d+1$ distinct sizes
	Nested top-k	like $\mathbf{h}_\emptyset$	the top-ranked memories' values	among equally relevant sets
	Private memories	inside each block	$\geq \sum_b \max(n_b - T_b, 0)$ values	share one store
	Tied memories	$\mathbf{e}_{\{i\}} - \mathbf{e}_{\{j\}}$	the split of $\varphi_i + \varphi_j$	merge, or retrieve apart

4 LEDGER: Valuing Memory Where the Log Allows

Section 3 showed that every undetermined value comes from an invisible change of the log, and that retrieval decides which invisible changes a log contains. This section presents LEDGER, a method built on one object, the invisible changes of the log: every value it reports is one that no invisible change moves, and each of its later actions removes invisible changes (Figure 2). With the log the agent already has, LEDGER first estimates each determined value with a confidence interval and reports every other value as undetermined. It then merges memories that are always retrieved together, which removes the invisible changes between them and keeps every determined value. For the logs the agent will collect, it removes the invisible changes that remain: it varies the retrieval size and fills part of each retrieval with the relevant memories retrieved least often. Appendix C gives the three actions as algorithms.

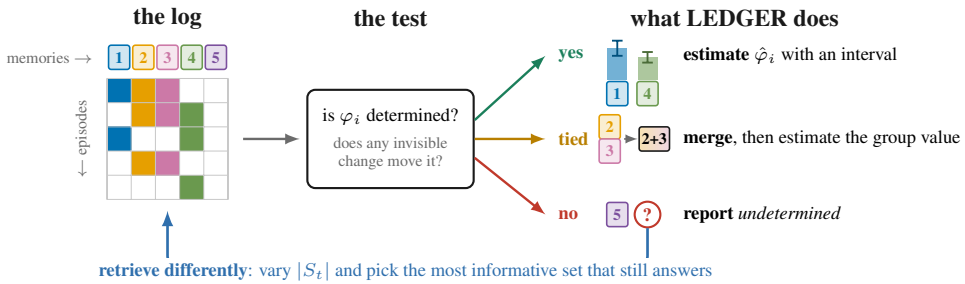

Figure 2: LEDGER reads the kernel of the log from its counts and acts on each memory accordingly: it estimates the values the log determines, merges memories the log cannot separate, abstains on the rest. In the log shown, memories 2 and 3 are always retrieved together.

Estimation. LEDGER checks the condition of Section 3 numerically (Algorithm 1): with $\mathbf{U}$ an orthonormal basis of $\ker \mathbf{X}_d$, memory i counts as determined when $\|\mathbf{U}^\top \mathbf{c}^{(i)}\| \le \tau \|\mathbf{c}^{(i)}\|$, with $\tau = 10^{-8}$. This check must come first, because a standard solver, such as least squares or ridge regression, returns a number for every memory, including undetermined ones (Lemma A.2). For an undetermined memory, that number estimates only the part of φ_i that no invisible change moves. For each determined memory, LEDGER estimates φ_i from a ridge fit with an interval,

$$\hat{\varphi}_i = (\mathbf{c}^{(i)})^\top \hat{\boldsymbol{\theta}}, \quad \hat{\boldsymbol{\theta}} = (\mathbf{X}_d^\top \mathbf{X}_d + \lambda \mathbf{I})^{-1} \mathbf{X}_d^\top \mathbf{r}, \qquad \hat{\varphi}_i \pm z_{1-\delta/(2n)}\, \hat{\sigma}\, \left\| \mathbf{X}_d (\mathbf{X}_d^\top \mathbf{X}_d + \lambda \mathbf{I})^{-1} \mathbf{c}^{(i)} \right\|, \quad (5)$$

where $\lambda = 10^{-2}$ and $\hat{\sigma}$ is the residual standard deviation. The interval rests on the following:

Proposition 3 (Intervals for determined values)**.** *Suppose the noise ε_t has mean zero and is σ-sub-Gaussian given the retrieved sets, and the retrieved sets do not depend on past outcomes. Then for each determined memory i, with probability at least $1 - \delta/n$,*

$$|\hat{\varphi}_i - \varphi_i| \;\le\; \sigma \sqrt{2 \log(2n/\delta)}\, \left\| \mathbf{X}_d \mathbf{V}^{-1} \mathbf{c}^{(i)} \right\| \;+\; \lambda \left| (\mathbf{c}^{(i)})^\top \mathbf{V}^{-1} \boldsymbol{\theta} \right|, \qquad \mathbf{V} = \mathbf{X}_d^\top \mathbf{X}_d + \lambda \mathbf{I}, \quad (6)$$

so the bound holds for all n memories at once with probability at least $1 - \delta$.

The first term is the width of Eq. (5), with the Gaussian quantile $z_{1-\delta/(2n)}$ in place of $\sqrt{2 \log(2n/\delta)}$; the second is the ridge bias, which vanishes as $\lambda \to 0$ for determined values and does not for undetermined ones (Appendix B.7). The retrieval rule of this section reads the past only through the retrieval counts, so it satisfies the second condition. Every other memory is reported as *undetermined*, and only determined values guide which memories to keep. Memories that are always retrieved together are among the undetermined ones, and the next action handles them.

Merging. When memories i and j are always retrieved together, the invisible change $\mathbf{e}_{\{i\}} - \mathbf{e}_{\{j\}}$ raises φ_i and lowers φ_j by the same amount, so the log can determine at most the pair's total value. LEDGER treats the pair as one memory. It groups memories whose columns of $\mathbf{X}$ are identical, which hashing the columns finds in $O(nT)$ time (Algorithm 2), and replaces each group g by one memory whose column is the shared one and whose value is the sum $\theta_g = \sum_{i \in g} \theta_{\{i\}}$; this gives a merged matrix $\bar{\mathbf{X}}$. Replacing memories changes the parameters, so the question is whether it changes what the log determines. Theorem 2 shows that every determined value is kept.

Theorem 2 (Merging keeps every determined value)**.** *At $d = 1$, $\operatorname{range} \mathbf{X}_1 = \operatorname{range} \bar{\mathbf{X}}_1$, so the two matrices give the same outcomes. A combination $\mathbf{c}^\top \boldsymbol{\theta}$ is determined by $\mathbf{X}_1$ exactly when $\mathbf{c}$ takes one value on each group and the corresponding combination of group values is determined by $\bar{\mathbf{X}}_1$.*

Every value determined before merging is determined after it, and every invisible change between tied memories is gone; the first action then estimates each group value that $\bar{\mathbf{X}}_1$ determines. The rule reads only retrieval histories: two unrelated memories with the same history are merged, and two near-duplicates retrieved in different episodes stay separate. Merging memories whose histories differ slightly is an approximation. For any grouping at $d = 1$, let a merged memory count as retrieved whenever one of its members is, let D_g be the largest number of episodes in which group g was retrieved while one of its members was not, let $s_g(\boldsymbol{\theta}) = \sum_{i \in g} \theta_{\{i\}}$, and let $\bar{\boldsymbol{\theta}} = (\theta_\emptyset, s_{g_1}(\boldsymbol{\theta}), \ldots, s_{g_m}(\boldsymbol{\theta}))$.

Theorem 3 (Merging memories with similar histories)**.** *For every combination $\mathbf{c}$ in the row space of $\bar{\mathbf{X}}_1$, the value that the merged log determines for $\mathbf{c}^\top \bar{\boldsymbol{\theta}}$ differs from it by at most*

$$\frac{\|\mathbf{c}\|}{\sigma^+_{\min}(\bar{\mathbf{X}}_1)} \sum_{i=1}^{n} |\theta_{\{i\}}| \sqrt{D_{g(i)}}, \quad (7)$$

where $\sigma^+_{\min}$ is the smallest nonzero singular value and $g(i)$ is the group of memory i. The bound is zero when every $D_g = 0$, which is Theorem 2, and neither its dependence on $\sqrt{D_g}$ nor on $1/\sigma^+_{\min}$ can be improved.

Replacing $|\theta_{\{i\}}|$ by the upper end of its interval from Eq. (5) makes the bound computable.

Retrieval. The next log needs a store shared across queries, at least $d + 1$ distinct retrieval sizes (Proposition 1), and sets that are not nested (Table 1). LEDGER keeps the store shared and draws

the size k_t of each retrieval from at least $d+1$ values. Let R_t be the M memories most similar to q_t, and let $f_i(t)$ be the number of episodes before t that retrieved memory i. LEDGER retrieves

$$S_t = L_t \cup H_t, \tag{8}$$

where L_t holds the $\max(1, \mathrm{round}(\rho k_t))$ memories of R_t with the smallest $f_i(t)$, and H_t holds the $k_t - |L_t|$ memories of R_t most similar to q_t (Algorithm 3). When L_t and H_t overlap, $|S_t| < k_t$. The memories in H_t keep the answer within reach. The memories in L_t target the least certain estimates: for small λ, the variance of $\hat{\theta}_{\{i\}}$ is at least $\sigma^2/(f_i(t)+\lambda)$, since $(\mathbf{V}^{-1})_{ii} \geq 1/V_{ii}$ for every positive definite $\mathbf{V}$ (Appendix A.5). Because $f_i(t)$ changes from pass to pass, one query retrieves different sets over time, which breaks the nesting of top-k. The baseline test (Definition A.6) checks from the log whether a change like $\mathbf{h}_\emptyset$ remains. We use $\rho = 0.4$ and $M = 20$. Because this rule changes which memories the reader sees, it trades answer quality for determined values.

5 Experiments

Sections 3 and 4 make four claims that only data can test. First, by Theorem 1 and Proposition 2, the logs of standard benchmarks leave every memory value undetermined; Section 5.2 checks this on fourteen benchmarks. Second, by Proposition 1 and the retrieval rule Eq. (8), LEDGER's retrieval makes many more values determined; Section 5.3 measures how many, and at what cost in accuracy. Third, by Proposition 3, the interval in Eq. (5) contains each value LEDGER reports at the stated rate. Fourth, by Theorem 2, merging keeps every determined value and makes the total value of each group of tied memories determinable. Section 5.4 tests both.

5.1 Setup

The audit (Section 5.2 and Appendix E) runs fourteen memory benchmarks, each with the retrieval it specifies. The retrieval experiments use three corpora with different structure: HotpotQA (Yang et al., 2018) and LongMemEval (Wu et al., 2025), whose questions we pool into one shared store per corpus, and LoCoMo (Maharana et al., 2024), whose questions already share a store within each conversation. Memories are turn- or paragraph-level spans, retrieved by dense similarity over sentence embeddings (Reimers & Gurevych, 2019). Answers come from a hosted reader, `gemini-2.5-flash-lite`, and are scored by exact match, with the same model as judge otherwise; two further readers, `gemini-3.6-flash` and `gemini-3.7-flash`, measure the cost of pooling a store, and they and the use of determined values to decide which memories to keep appear in Appendix F. We compute every identifiability number from $\mathbf{X}$ alone, and compare retrieval rules on the same questions, reporting each accuracy difference with a 95% interval and a paired permutation test. Coverage of the intervals is measured on synthetic logs (Appendix D.4). Table 4 names the artifact behind each number below.

5.2 What standard benchmarks determine

For every memory, we checked whether $\mathbf{c}^{(i)}$ lies in the row space of $\mathbf{X}_1$, on the logs of fourteen memory benchmarks, each run with the retrieval it specifies and at every k it uses (Table 2). We ran fourteen benchmarks, each with its own retrieval at every k it uses, and tested every memory at $d = 1$. Two conventions cause this. Within each block of the log, every episode retrieves the same number of memories (Theorem 1). Most benchmarks also give each question or document its own store, which splits the log into many blocks (Proposition 2). A block is a group of memories never retrieved with a memory outside it, and even a shared store splits into blocks. Block counts follow the order of k, and "all" retrieves every memory of the context.

All 55 configurations leave every memory value undetermined, for two reasons. Within each context, every episode retrieves the same number of memories, so a change like $\mathbf{h}_\emptyset$ is invisible in every log (Theorem 1). Twelve of the fourteen benchmarks also give each question, document or conversation its own memories, which splits the log into blocks (Proposition 2). The two benchmarks with one shared store, MultiHopRAG and HELMET, leave every value undetermined because their retrieval size is fixed. LoCoMo shares a store within each conversation and splits its $1{,}542$ questions into 13 blocks at $k = 5$, with the same result. A shared store is therefore necessary and not sufficient. Figure 3a compares LEDGER's retrieval rule Eq. (8) with deterministic top-k and with two random

rules, ε-greedy and score perturbation, on pooled HotpotQA. The fraction of memories determined is measured on 300 questions retrieved over 24 passes ($n = 2{,}971$ memories, $T = 7{,}200$ episodes), and accuracy on 1,600 held-out questions against a pooled store of 15,900 memories.

Table 2: No memory benchmark determines a single memory value.

Benchmark	k	Store	T	n	Blocks	Determined
HotpotQA	3, 5, 10	per question	300	2,971	300	0
HotpotQA (1,200 q.)	3, 5, 10	per question	1,200	11,912	1,200	0
MuSiQue	3, 5, 10	per question	300	5,999	300	0
WikiHop	3, 5, 10	per question	300	4,560	300	0
TriviaQA	3, 5, 10	per question	300	27,456	300	0
LongMemEval	3, 5, 10	per question	300	6,845	300	0
MSC-Self-Instruct	3, 5, 10	per dialogue	300	18,916	300	0
BABILong	3, 5, 10, all	per item	40	4,158	40	0
InfiniteBench	3, 5, 10, all	per book	40	37,697	40	0
NarrativeQA (summary)	3, 5, 10, all	per document	300	115	11	0
NarrativeQA (full)	3, 5, 10, all	per document	300	14,282	121, 60, 27, 11	0
QuALITY	3, 5, 10, all	per article	219	1,495	47, 22, 12, 12	0
LoCoMo	3, 5, 10	per conversation	1,542	5,882	37, 13, 11	0
MemoryAgentBench	3, 5, 10	per context	360	33,686	94, 65, 44	0
MultiHopRAG	3, 5, 6, 10	shared	2,556	23,078	269, 98, 66, 15	0
HELMET	3, 5, 10, 14	shared	356	40,000	41	0

Table 2 tests every memory on 55 configurations of the 14 benchmarks, at $d = 1$ and therefore at every resolution (Lemma B.3). The first reason is the retrieval size. In every log, all episodes of a context retrieve the same number of memories, k or the whole context, so the log contains a change like $\mathbf{h}_\emptyset$ that reaches every memory (Theorem 1, and Lemma E.1 when the size differs between contexts). The second reason is the store. Twelve of the fourteen benchmarks give each question, document or conversation its own memories, which splits the log into blocks (Proposition 2). In HotpotQA, for example, each question has about ten memories of its own and is asked once, so varying the size would determine at most one of its ten values. A shared store also splits: MultiHopRAG's 2,556 questions fall into 269 blocks at $k = 3$ and 15 at $k = 10$, and HELMET's 356 into 41. A shared store is necessary and not sufficient.

5.3 Retrieval and its cost

Hence, we vary the retrieval size on a shared store. On pooled HotpotQA, four rules draw the size from $\{3, 5, 8\}$ (Figure 3a): deterministic top-k, LEDGER's rule in Eq. (8), ε greedy and score perturbation. LEDGER's sets fall below the drawn size in 6.7% of episodes, when L_t and H_t overlap (Appendix D). The fraction of memories determined is measured on 300 questions retrieved over 24 passes ($n = 2{,}971$ memories, $T = 7{,}200$ episodes). Accuracy is measured on 1,600 held-out questions against a pooled store of 15,900 memories. The accuracy of the random rules is measured on 300 questions, with Holm corrected intervals. Deterministic top-k determines 0.001 of the memories, although its size varies. The reason is nesting: for each question, it retrieves the top of one ranking, so the set at each size contains the smaller ones. In this log, $\mathbf{X}\mathbf{w} = \mathbf{1}_T$ has a least-squares solution with residual 3.1×10^{-15}. Lowering the baseline by s and raising each $\theta_{\{i\}}$ by $s\, w_i$ then leaves every predicted outcome unchanged, an invisible change ($\mathbf{h}_\emptyset$). Pooling the store has its own cost, which this comparison does not include: on LongMemEval, answering from a pooled store costs 15.7 points against each question's own store (Appendix F.4).

LEDGER's rule fills part of each set with the relevant memories retrieved least often, and these change from pass to pass, which breaks the nesting. Its residual is 9.2×10^{-2}, so $\mathbf{1}_T$ lies outside the range of $\mathbf{X}$ and this change is gone. The fraction determined rises to 0.797, at a cost of 2.6 accuracy points (95% interval $[-4.2, -0.9]$, permutation $p = 0.004$). The random rules trade worse. At about LEDGER's cost, ε-greedy with $\varepsilon = 0.1$ loses 3.0 points and determines 0.001. To determine 0.98 or more, the random rules lose at least 13.3 points.

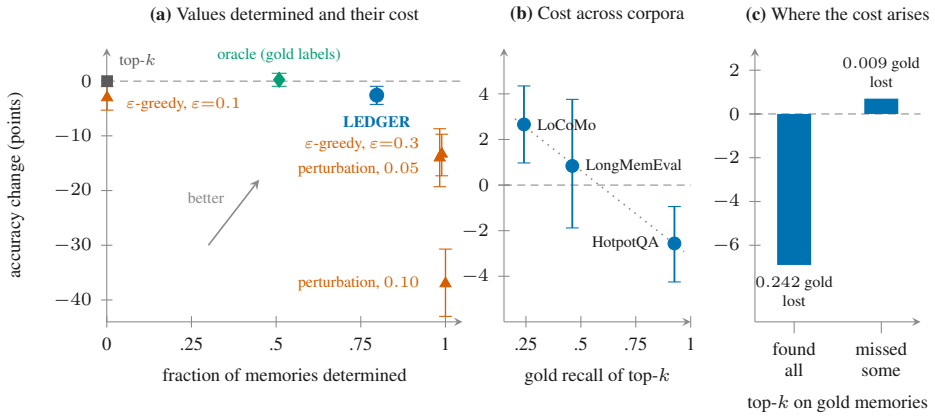

Figure 3: What LEDGER's retrieval determines, what it costs, and where the cost arises.

LEDGER's effect on accuracy depends on how much evidence top-k already finds. We call a memory *gold* when it holds evidence for the answer. On HotpotQA, LongMemEval and LoCoMo, top-k retrieves 0.93, 0.46 and 0.24 of the gold memories, and LEDGER changes accuracy by -2.6, $+0.8$ ($[-1.9, +3.8]$) and $+2.7$ points ($[+1.0, +4.4]$). Figure 3b shows the mechanism on HotpotQA. LEDGER replaces some of the similar memories with less-retrieved ones, which can displace gold memories that top-k had found. On questions where top-k retrieved every gold memory, LEDGER drops 0.242 of them per question and loses 6.9 points. On the other questions, it drops 0.009 and gains 0.7. An oracle that applies LEDGER only to the questions where top-k missed a gold memory removes the cost ($[-0.94, +1.44]$). It determines 0.509 of the memories, 64% of LEDGER's gain.

5.4 Intervals and Merging

Checking whether the intervals are valid requires the true values, so we use synthetic logs drawn from the model Eq. (2) with known $\boldsymbol{\theta}$, over 36 settings of n, k, T and λ at each resolution, with 20 random designs per setting (Appendix D). A log counts as covered when the interval in Eq. (5), with the noise scale known, contains φ_i for every memory LEDGER reports, and at $\delta = 0.05$ at least 95% of logs should be covered. At $d = 1$, every design has full rank, and at least 97.7% of logs are covered in every setting. At $d = 2$, 24 settings give rank-deficient designs, and at least 99.4% of logs are covered in all 36 settings, in a post hoc analysis (Appendix G.3). Counting every memory instead, coverage falls to 57.2% in the worst setting, but every memory there is undetermined. The intervals are 1.52 and 1.43 times as wide as the pointwise Gaussian interval with the true σ at $d = 1$ and $d = 2$. The confidence set of Abbasi-Yadkori et al. (2011) is a median 6.6 and 36 times as wide at $\lambda = 1$, because it bounds every direction of $\boldsymbol{\theta}$, while LEDGER bounds only the n directions $\mathbf{c}^{(i)}$ that define the memory values.

Merging replaces tied memories by groups. The value of each tied memory is undetermined, but the total value of its group can be determined, and by Theorem 2 every other memory keeps its status. We therefore count units, meaning memories and groups. On pooled HotpotQA with 100 questions and LEDGER's retrieval ($n = 992$ memories, $T = 2{,}400$), merging identical histories leaves 869 units and raises the fraction determined from 0.815 to 0.992. Merging also helps on conversational memory. On two LoCoMo conversations ($n = 788$ memories), the standard log determines no value (Table 2). Three passes of retrieval with score perturbation determine 79.0% of their memories, and merging raises this to 95.7% of the units.

6 Conclusion

A retrieval log determines a memory's value only if the retrievals separate that memory from the others, and this can be checked from the log before any outcome is seen. Standard benchmarks fail this check: a fixed retrieval size and private stores leave every memory value undetermined in all 55 configurations of fourteen benchmarks. Both causes are choices about retrieval, so changing retrieval removes them. LEDGER raises the fraction of memories determined from 0.001 to 0.797 at a cost

of 2.6 accuracy points. Every value LEDGER reports is determined by the log and comes with a confidence interval, and every other value is reported as undetermined. Memory benchmarks should report which values their logs determine alongside accuracy. Finding at query time the questions where spreading retrieval costs nothing is the open problem that remains.

AI USE STATEMENT

Generative AI tools helped design experiments and choose their parameters, generate synthetic data, write code, and clean datasets. On the literature, they helped search, summarize, and analyze prior work and format references. On the paper, they helped create figures and tables, draft text, and edit it. We reviewed all AI-assisted work and checked it in four ways. Every empirical claim was pre-registered with a decision rule, and three claims that failed their rule were withdrawn (Appendix G). Every reported number is reproduced from committed artifacts and was reconciled against them in a separate audit. Every theoretical claim was tested on synthetic designs with known ground truth. Every bibliography entry was verified against its source. We take full responsibility for the content of this paper.

ETHICS STATEMENT

This work involves no human subjects, no new data collection, and no release of new data, so no IRB review was required. All experiments use public benchmarks under their published licenses; their contents are encyclopedic text and synthetically generated dialogue, and we neither collected nor inferred personal information from them. Two consequences of the method deserve explicit statement. First, the instrumentation we recommend is not free to the end user: an A-optimal tie-break deliberately selects among answer-preserving retrieval sets by information gain, and we measure an accuracy cost whose sign depends on the corpus, so a practitioner adopting it trades some answer quality for the ability to value memory at all. We report both signs, the mechanism that produces them (Section 4), and a restriction that removes the cost on the corpus where it is largest, so that this trade is made deliberately and not inherited from a default. Second, a framework that makes the value of an individual stored memory identifiable thereby makes attribution to that memory identifiable. The same quantity that justifies retaining or deleting a record supports profiling the interactions the record was derived from. We declare no conflicts of interest. On research integrity, claims here are pre-registered, null and negative results are reported alongside positive ones (Appendix F), and corrections to values that appeared in earlier drafts are recorded in the paper.

REPRODUCIBILITY STATEMENT

The two modeling assumptions are stated in Section 2 and their scope is discussed in Appendix A.3; every theoretical claim in Sections 3 to 4 is proved in full in Appendix B, with the merge-map construction and its complexity given as explicit algorithms in Appendix C; the synthetic generator used to check those claims against known ground truth is specified in Appendix D, and the tables reporting every headline number are generated from committed artifacts and collected in Appendix F. Anonymous source code, the dated pre-registration documents, the exact benchmark preparation and store-construction steps, and the artifacts from which every table is generated are available at `https://anonymous.4open.science/r/ledger-review-AF71/`. The theoretical track (identifiability verdicts, ranks, the exact quotient, and all synthetic experiments) is fully reproducible and deterministic from the released code and a fixed seed, since it consults no model. The empirical track is reproducible in its design and not claimed to be bit-reproducible in its outcomes: the retrieval logs, design matrices and identifiability measurements regenerate byte-identically from the released inputs, while the reader accuracies depend on hosted model inference that we cannot pin, so we report paired comparisons with intervals and the sample sizes they were computed at. Appendix G documents the defects we found in our own pipeline, the values they affected, and the corrected values, so that a reader attempting to reproduce this work knows which numbers changed and why.

REFERENCES

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Filip Radlinski, and Thorsten Joachims. Estimating position bias without intrusive interventions. In *ACM International Conference on Web Search and Data Mining (WSDM)*, 2019.

John R Anderson and Lael J Schooler. Reflections of the environment in memory. *Psychological science*, 2(6):396–408, 1991.

Mustafa Arslan. Aeon: High-performance neuro-symbolic memory management for long-horizon LLM agents. *arXiv preprint arXiv:2601.15311*, 2026. URL https://arxiv.org/abs/2601.15311.

John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19: 317, 1964.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.

Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z Gajos. To trust or to think: cognitive forcing functions can reduce overreliance on ai in ai-assisted decision-making. *Proceedings of the ACM on Human-computer Interaction*, 5(CSCW1):1–21, 2021.

Cheng Chen, Maria D Molina, Mengqi Liao, and Eugene Cho Snyder. Relational gains, privacy strains: Exploring users' perceptions and experiences with chatgpt's memory feature. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2026a.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. *International Conference on Machine Learning (ICML)*, 2013.

Yanqiao Chen, Dongsheng Hou, Yuhan Rui, Zhen Cao, and Yepang Liu. Shapley context pruning: A cooperative game perspective for context reranking and pruning. *arXiv preprint arXiv:2607.16209*, 2026b. URL https://arxiv.org/abs/2607.16209.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Mądry. Contextcite: Attributing model generation to context. *Advances in Neural Information Processing Systems*, 37:95764–95807, 2024.

Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm. *Discrete Applied Mathematics*, 7(3):251–274, 1984.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z. Pan. Rethinking memory in LLM based agents: Representations, operations, and emerging topics. *arXiv preprint arXiv:2505.00675*, 2025. URL https://arxiv.org/abs/2505.00675.

Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Hammer. SHAP-IQ: Unified approximation of any-order shapley interactions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.01179.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning (ICML)*, 2019.

Manu Ghulyani, Arunabh Singh, Karan Bharadwaj, Ankit Nath, and Suranjan Goswami. PACMS: Submodular context selection as a pluggable engine for LLM agents. *arXiv preprint arXiv:2606.20047*, 2026. URL https://arxiv.org/abs/2606.20047.

Michel Grabisch, Jean-Luc Marichal, and Marc Roubens. Equivalent representations of set functions. *Mathematics of Operations Research*, 25(2):157–178, 2000.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Xiaoxue Han, Zhuo Feng, and Yue Ning. A topology-aware graph coarsening framework for continual graph learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ef283d62b4bce30854a8d4827f331229-Abstract-Conference.html.

John C. Harsanyi. A simplified bargaining model for the n-person cooperative game. *International Economic Review*, 4(2):194–220, 1963.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/efaf1c9726648c8ba363a5c927440529-Abstract-Conference.html.

Yudai Honda, Yuki Fujita, Keiichi Zempo, and Shogo Fukushima. Human-like remembering and forgetting in LLM agents: An ACT-R-inspired memory architecture. In *Proceedings of the 13th International Conference on Human-Agent Interaction (HAI)*, 2025. doi: 10.1145/3765766.3765803.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, volume 2026, pp. 156259–156291, 2026.

Yu Huo, Kun Zeng, Siyu Zhang, Yuquan Lu, Cheng Yang, Yifu Guo, and Xiaoying Tang. RepoShapley: Shapley-enhanced context filtering for repository-level code completion. *arXiv preprint arXiv:2601.03378*, 2026. URL https://arxiv.org/abs/2601.03378.

Xinke Jiang, Rihong Qiu, Yongxin Xu, Wentao Zhang, Yichen Zhu, Ruizhe Zhang, Yuchen Fang, Xu Chu, Junfeng Zhao, and Yasha Wang. RAGraph: A general retrieval-augmented graph learning framework. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/34d6c7090bc5af0b96aeaf92fa074899-Abstract-Conference.html.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/6ddc001d07ca4f319af96a3024f6dbd1-Abstract-Conference.html.

Antonin Joly and Nicolas Keriven. Graph coarsening with message-passing guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d041d6bb47c01a4ce327a1773703e9a0-Abstract-Conference.html.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Justin Singh Kang, Landon Butler, Abhineet Agarwal, Yigit Efe Erginbas, Ramtin Pedarsani, Kannan Ramchandran, and Bin Yu. SPEX: Scaling feature interaction explanations for LLMs. *arXiv preprint arXiv:2502.13870*, 2025. URL https://arxiv.org/abs/2502.13870.

Haruka Kiyohara, Masahiro Nomura, and Yuta Saito. Off-policy evaluation of slate bandit policies via optimizing abstraction. *arXiv preprint arXiv:2402.02171*, 2024. URL https://arxiv.org/abs/2402.02171.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554, 2024.

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

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

Qinfeng Li, Yuntai Bao, Xinyan Yu, Hongze Chen, Wenqi Zhang, and Xuhong Zhang. AttriMem: Attribution-guided process feedback for agent memory learning. *arXiv preprint arXiv:2607.21106*, 2026. URL https://arxiv.org/abs/2607.21106.

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. StructRAG: Boosting knowledge intensive reasoning of LLMs via inference-time hybrid information structurization. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://iclr.cc/virtual/2025/poster/30265.

Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. HopRAG: Multi-hop reasoning for logic-aware retrieval-augmented generation. In *Findings of the Association for Computational Linguistics (ACL)*, 2025. URL https://aclanthology.org/2025.findings-acl.97/.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.01061.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://iclr.cc/virtual/2025/poster/28367.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2402.17753.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, et al. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5336–5358, 2022.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Sara Patel, Mingxun Zhou, and Giulia Fanti. MaxShapley: Towards incentive-compatible generative search with fair context attribution. *arXiv preprint arXiv:2512.05958*, 2025. URL https://arxiv.org/abs/2512.05958.

Mathis Pink, Qinyuan Wu, Vy A. Vo, Javier Turek, Jianing Mu, Alexander Huth, and Mariya Toneva. Position: Episodic memory is the missing piece for long-term LLM agents. *arXiv preprint arXiv:2502.06975*, 2025. URL https://arxiv.org/abs/2502.06975.

Friedrich Pukelsheim. *Optimal Design of Experiments*. SIAM, 2006.

C. Radhakrishna Rao. *Linear Statistical Inference and Its Applications*. Wiley, 2nd edition, 1973.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, 2019. URL https://aclanthology.org/D19-1410/.

Idan Rejwan and Yishay Mansour. Top-k combinatorial bandits with full-bandit feedback. In *Algorithmic Learning Theory (ALT)*, 2020. URL https://arxiv.org/abs/1905.12624.

Paul R. Rosenbaum and José R. Zubizarreta. Effect aliasing in observational studies. *Journal of the American Statistical Association*, 2025. doi: 10.1080/01621459.2025.2537456. Published online; arXiv:2408.16708.

Max Schemmer, Niklas Kuehl, Carina Benz, Andrea Bartos, and Gerhard Satzger. Appropriate reliance on ai advice: Conceptualization and the effect of explanations. In *Proceedings of the 28th international conference on intelligent user interfaces*, pp. 410–422, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, 1953.

Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miroslav Dudík, John Langford, Damien Jose, and Imed Zitouni. Off-policy evaluation for slate recommendation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1605.04812.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. URL https://aclanthology.org/2025.acl-long.413/.

Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*, 2024.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Che-Ping Tsai, Chih-Kuan Yeh, and Pradeep Ravikumar. Faith-shap: The faithful shapley interaction index. *Journal of Machine Learning Research*, 24, 2023. URL https://jmlr.org/papers/v24/22-0202.html.

Robert J Weber. Probabilistic values for games. *The Shapley Value. Essays in Honor of Lloyd S. Shapley*, pp. 101–119, 1988.

Lei Wei, Xiao Peng, Xu Dong, Niantao Xie, and Bin Wang. FadeMem: Biologically-inspired forgetting for efficient agent memory. *arXiv preprint arXiv:2601.18642*, 2026a. URL https://arxiv.org/abs/2601.18642.

Shuyue Wei, Chang Liu, Zimu Zhou, Yongxin Tong, and Lizhen Cui. MemLens: A value-aware memory management system with interactive analytics for LLM-based agents. *arXiv preprint arXiv:2607.25992*, 2026b. URL https://arxiv.org/abs/2607.25992.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018.

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*, 2026a. URL https://arxiv.org/abs/2606.25115.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.10813.

Zhaofen Wu, Hanrong Zhang, Fulin Lin, Wujiang Xu, Xinran Xu, Yankai Chen, Henry Peng Zou, Shaowen Chen, Weizhi Zhang, Xue Liu, Philip S. Yu, and Hongwei Wang. GAM: Hierarchical graph-based agentic memory for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026b. URL https://aclanthology.org/2026.acl-long.1600/.

J Xu, A Szlam, and J Weston. Beyond goldfish memory: Long-term open-domain conversation. arxiv 2021. *arXiv preprint arXiv:2107.07567*, 2021.

Sikuan Yan, Ahmed Bahloul, Ercong Nie, Susanna Schwarzmann, Riccardo Trivisonno, Volker Tresp, and Yunpu Ma. Memory-r2: Fair credit assignment for long-horizon memory-augmented LLM agents. *arXiv preprint arXiv:2605.21768*, 2026a. URL https://arxiv.org/abs/2605.21768.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, et al. 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, 2026b.

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.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. URL https://aclanthology.org/D18-1259/.

Zikun Ye and Hema Yoganarasimhan. Document valuation in LLM summaries: A cluster shapley approach. *arXiv preprint arXiv:2505.23842*, 2025. URL https://arxiv.org/abs/2505.23842.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context models effectively and thoroughly. In *International Conference on Learning Representations*, volume 2025, pp. 98914–98965, 2025.

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.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. Bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

Qiming Zhu, Shunian Chen, Rui Yu, Zhehao Wu, and Benyou Wang. From lossy to verified: A provenance-aware tiered memory for agents. *arXiv preprint arXiv:2602.17913*, 2026. URL https://arxiv.org/abs/2602.17913.

Appendix

A Preliminaries and Related Work **18**

A.1 Linear models and estimability 18

A.2 Set functions, dividends and values 18

A.3 Assumptions and their scope 19

A.4 The design of a retrieval log 20

A.5 Optimal design and confidence sets 21

A.6 Related work 22

B Proofs **23**

B.1 Counts decide identifiability 24

B.2 A fixed retrieval size 24

B.3 Sizes needed 24

B.4 Nested sets 25

B.5 Private memories 26

B.6 Resolution 26

B.7 Intervals 26

B.8 Merging 27

C Algorithms **28**

C.1 The test and the estimates 28

C.2 Merging 28

C.3 Retrieval 29

C.4 The baseline test and costs 29

D Experimental Setup **30**

D.1 Corpora and stores 30

D.2 Retrieval and reading 30

D.3 Retrieval rules 31

D.4 Synthetic logs 31

D.5 Statistics 31

D.6 Runs behind the main text 31

E Benchmark Audit **32**

E.1 Protocol 32

E.2 Why every configuration determines nothing 33

E.3 The benchmarks 33

F Additional Results **33**

F.1 Retrieval on pooled HotpotQA 33
F.2 Replications on LoCoMo and LongMemEval 34
F.3 The mechanism of the accuracy cost 35
F.4 Pooling across readers 35
F.5 Intervals 35
F.6 Merging 36
F.7 Uses of the determined values 36

G Pre-registration and Corrections **37**
G.1 Pre-registered claims 37
G.2 Withdrawn claims 37
G.3 The post hoc analysis 37
G.4 Amendment on the retrieval rule 37
G.5 Errata 38

H Discussion **38**
H.1 Interpreting the results 38
H.2 Scope and open problems 40
H.3 Applications 41
H.4 Limitations 42

I Notation and Terms **43**

A Preliminaries and Related Work

This appendix collects the classical results the paper builds on and fixes the quantities it reports. Appendix A.1 states when a linear model determines a linear functional, Appendix A.2 writes memory values as such functionals, and Appendix A.3 discusses the scope of the two assumptions. Appendix A.4 applies these results to a retrieval log, Appendix A.5 gives the optimal-design and confidence-set results that Sections 4 and 5 use, and Appendix A.6 relates the work to the literature. Notation and terms are collected in Appendix I.

A.1 Linear models and estimability

Consider the linear model $\mathbf{r} = \mathbf{A}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ with a known matrix $\mathbf{A} \in \mathbb{R}^{T\times p}$, an unknown parameter $\boldsymbol{\beta} \in \mathbb{R}^p$ and $\mathbb{E}\,\boldsymbol{\varepsilon} = \mathbf{0}$. The question is which linear functionals $\mathbf{c}^\top\boldsymbol{\beta}$ the data determine.

Definition A.1 (Estimable functional)**.** A linear functional $\mathbf{c}^\top\boldsymbol{\beta}$ is *estimable* if some $\mathbf{a} \in \mathbb{R}^T$ satisfies $\mathbb{E}[\mathbf{a}^\top\mathbf{r}] = \mathbf{c}^\top\boldsymbol{\beta}$ for every $\boldsymbol{\beta} \in \mathbb{R}^p$.

Lemma A.1 (Rao, 1973, Section 4a.2)**.** *$\mathbf{c}^\top\boldsymbol{\beta}$ is estimable if and only if $\mathbf{c} \in \operatorname{range}\mathbf{A}^\top$, equivalently $\mathbf{c}^\top\mathbf{h} = 0$ for every $\mathbf{h} \in \ker\mathbf{A}$. If it is estimable, $\mathbf{c}^\top\hat{\boldsymbol{\beta}}$ takes the same value for every least-squares solution $\hat{\boldsymbol{\beta}}$, and when $\operatorname{Cov}\boldsymbol{\varepsilon} = \sigma^2\mathbf{I}$ this value is the best linear unbiased estimator of $\mathbf{c}^\top\boldsymbol{\beta}$.*

The first statement follows in one line: $\mathbb{E}[\mathbf{a}^\top\mathbf{r}] = (\mathbf{A}^\top\mathbf{a})^\top\boldsymbol{\beta}$ equals $\mathbf{c}^\top\boldsymbol{\beta}$ for every $\boldsymbol{\beta}$ exactly when $\mathbf{A}^\top\mathbf{a} = \mathbf{c}$. The equivalence with the kernel condition is $\operatorname{range}\mathbf{A}^\top = (\ker\mathbf{A})^\perp$. Two further facts are used throughout. First, $\ker\mathbf{A} = \ker(\mathbf{A}^\top\mathbf{A})$: if $\mathbf{A}^\top\mathbf{A}\mathbf{h} = \mathbf{0}$, then $\|\mathbf{A}\mathbf{h}\|^2 = \mathbf{h}^\top\mathbf{A}^\top\mathbf{A}\mathbf{h} = 0$, and the converse is immediate. Estimability is therefore decided by the Gram matrix $\mathbf{A}^\top\mathbf{A}$ alone. Second, a vector $\mathbf{h} \in \ker\mathbf{A}$ is invisible in the data: $\boldsymbol{\beta}$ and $\boldsymbol{\beta} + s\mathbf{h}$ give the same distribution of $\mathbf{r}$ for every $s \in \mathbb{R}$, while $\mathbf{c}^\top\boldsymbol{\beta}$ changes by $s\,\mathbf{c}^\top\mathbf{h}$. The next lemma states what a solver reports for a functional that is not estimable.

Lemma A.2 (What a solver reports)**.** *Write $\mathbf{c} = \mathbf{c}_1 + \mathbf{c}_2$ with $\mathbf{c}_1 \in \operatorname{range}\mathbf{A}^\top$ and $\mathbf{c}_2 \in \ker\mathbf{A}$. The least-squares solutions are $\mathbf{A}^\dagger\mathbf{r} + \ker\mathbf{A}$, where $\mathbf{A}^\dagger$ is the Moore–Penrose pseudoinverse. If $\mathbf{c}_2 \neq \mathbf{0}$, then $\mathbf{c}^\top\hat{\boldsymbol{\beta}}$ takes every real value as $\hat{\boldsymbol{\beta}}$ ranges over these solutions. The minimum-norm solution and the ridge solution $(\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^\top\mathbf{r}$, for every $\lambda > 0$, lie in $\operatorname{range}\mathbf{A}^\top$, so each reports $\mathbf{c}_1^\top\hat{\boldsymbol{\beta}}$: an estimate of $\mathbf{c}_1^\top\boldsymbol{\beta}$, which differs from $\mathbf{c}^\top\boldsymbol{\beta}$ by $\mathbf{c}_2^\top\boldsymbol{\beta}$.*

Proof. If $\hat{\boldsymbol{\beta}}$ is a least-squares solution, so is $\hat{\boldsymbol{\beta}} + s\mathbf{c}_2$ for every s, and $\mathbf{c}^\top(\hat{\boldsymbol{\beta}} + s\mathbf{c}_2) = \mathbf{c}^\top\hat{\boldsymbol{\beta}} + s\|\mathbf{c}_2\|^2$ takes every real value. The minimum-norm solution $\mathbf{A}^\dagger\mathbf{r}$ lies in $\operatorname{range}\mathbf{A}^\top$ by definition of the pseudoinverse, and the ridge solution equals $\mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top + \lambda\mathbf{I})^{-1}\mathbf{r}$, which lies there as well. For any $\hat{\boldsymbol{\beta}} \in \operatorname{range}\mathbf{A}^\top$, $\mathbf{c}_2^\top\hat{\boldsymbol{\beta}} = 0$. □

The reported number and its standard error look the same whether or not $\mathbf{c}_2 = \mathbf{0}$, so nothing in the output shows that the part $\mathbf{c}_2^\top\boldsymbol{\beta}$ was set to zero by the solver. LEDGER therefore tests $\mathbf{c}_2 = \mathbf{0}$ before it estimates (Section 4).

Lemma A.3 (Appending a column)**.** *For $\mathbf{A} \in \mathbb{R}^{T\times p}$ and $\mathbf{a} \in \mathbb{R}^T$, $\operatorname{rank}[\mathbf{a} \mid \mathbf{A}] = \operatorname{rank}\mathbf{A} + \mathbf{1}[\mathbf{a} \notin \operatorname{range}\mathbf{A}]$. Appending columns to $\mathbf{A}$ never makes a functional of the original parameters estimable that was not estimable before.*

Proof. The first statement is the definition of rank. For the second, let $\mathbf{h} \in \ker\mathbf{A}$ with $\mathbf{c}^\top\mathbf{h} \neq 0$. Then $(\mathbf{h}, \mathbf{0}) \in \ker[\mathbf{A} \mid \mathbf{B}]$ for any appended block $\mathbf{B}$, and its inner product with $(\mathbf{c}, \mathbf{0})$ is $\mathbf{c}^\top\mathbf{h} \neq 0$, so $(\mathbf{c}, \mathbf{0})$ is not estimable by Lemma A.1. □

A.2 Set functions, dividends and values

The results of Appendix A.1 apply to memory values once each value is a linear functional of a parameter vector. Möbius dividends provide that vector.

Definition A.2 (Möbius dividends; Harsanyi, 1963; Grabisch et al., 2000)**.** For $v : 2^{[n]} \to \mathbb{R}$, the Möbius dividend of $B \subseteq [n]$ is $\theta_B = \sum_{A\subseteq B}(-1)^{|B\setminus A|}\,v(A)$. Then $v(S) = \sum_{B\subseteq S}\theta_B$ for every $S \subseteq [n]$, and the dividends are the unique numbers with this property. We call v *d-additive* if $\theta_B = 0$ whenever $|B| > d$.

The dividend θ_B is the part of v that the members of B create jointly and no proper subset of B creates. The dividend of the empty set is $\theta_\emptyset = v(\emptyset)$, the baseline. The Shapley value (Shapley, 1953) of element i is

$$\varphi_i(v) = \sum_{S \subseteq [n] \setminus \{i\}} \frac{|S|!\,(n-|S|-1)!}{n!} \big[v(S \cup \{i\}) - v(S)\big],$$

the average marginal contribution of i over all orders in which the elements can be added.

Lemma A.4 (Harsanyi, 1963)**.** *$\varphi_i = \sum_{B \ni i} \theta_B/|B|$. If v is d-additive, then $\varphi_i = (\mathbf{c}^{(i)})^\top \boldsymbol{\theta}$, where $\boldsymbol{\theta} = (\theta_B)_{|B| \le d}$ and $c_B^{(i)} = \mathbf{1}[i \in B]/|B|$. The baseline $\theta_\emptyset$ does not enter φ_i.*

Each dividend is thus shared equally among its members. Other values are linear in the dividends with other coefficients. The Banzhaf value (Banzhaf III, 1964), which averages the marginal contribution of i over all subsets with equal weight, is $\sum_{B \ni i} 2^{1-|B|}\theta_B$. More generally, every probabilistic value (Weber, 1988) averages the marginal contributions $v(S \cup \{i\}) - v(S) = \sum_{A \subseteq S} \theta_{A \cup \{i\}}$ with some weights, so it is linear in $\boldsymbol{\theta}$, and the estimability test of Appendix A.1 applies to it with its own coefficient vector. At $d = 1$ every marginal contribution equals $\theta_{\{i\}}$, so every probabilistic value of i, the Shapley value included, equals $\theta_{\{i\}}$.

A.3 Assumptions and their scope

Section 2 states two assumptions. This subsection says what each requires, what is lost when it fails, and the further condition that estimation needs.

Assumption 1. The assumption restricts neither the language model, the encoder nor the query distribution. It states that the expected outcome of an episode is a function of the query and the retrieved set, which is what it means for retrieval to matter, and that the deviations from this function are conditionally sub-Gaussian, which holds for every bounded outcome. The outcome may be binary or graded. Decisions about the whole store, such as what to keep and what to merge, need the query-averaged value

$$v(S) \;=\; \mathbb{E}_{q \sim \mathcal{D}_Q}\big[v_q(S)\big], \tag{9}$$

and the paper values memories under this v.

Assumption 2. The Shapley value is defined for every v, and Lemma A.4 writes it in dividends of every order. The resolution d keeps the dividends of order at most d, and the next proposition gives exactly what the others contribute.

Proposition A.1 (Truncation error)**.** *For any v and any d, let $v^{(d)}(S) = \sum_{B \subseteq S,\, |B| \le d} \theta_B$. Then*

$$\varphi_i(v) - \varphi_i\big(v^{(d)}\big) \;=\; \sum_{B \ni i,\ |B| > d} \frac{\theta_B}{|B|}, \qquad \textit{so} \qquad \big|\varphi_i(v) - \varphi_i(v^{(d)})\big| \le \frac{1}{d+1} \sum_{B \ni i,\, |B| > d} |\theta_B|. \tag{10}$$

Proof. The dividends of $v^{(d)}$ are those of v of order at most d and zero otherwise, by uniqueness in Definition A.2. Subtracting the two expressions of Lemma A.4 gives the identity, and $|B| \ge d+1$ gives the bound. □

The proposition needs no assumption on v. When Assumption 2 fails, the omitted dividends also enter the noise ε_t, and the estimates at resolution d are biased wherever these dividends correlate with the design. A memory undetermined at resolution d remains undetermined at every higher resolution (Appendix H.1), so the negative results of Section 5 hold whatever the true order of v.

Exogeneity. Write $v_q = v + \delta_q$, so that $\mathbb{E}_q\, \delta_q(S) = 0$ for every S. The noise of Eq. (2) is then $\varepsilon_t = \eta_t + \delta_{q_t}(S_t)$. Retrieval reads q_t when it chooses S_t, so $\delta_{q_t}(S_t)$ can correlate with the design.

Definition A.3 (Exogeneity)**.** A log is *exogenous* at resolution d if $\mathbb{E}\big[\mathbf{X}_d^\top \boldsymbol{\delta}\big] = \mathbf{0}$, where $\boldsymbol{\delta}_t = \delta_{q_t}(S_t)$.

A log is exogenous, for example, when S_t is independent of q_t: then $\mathbb{E}[\mathbf{1}[B \subseteq S_t]\,\delta_{q_t}(S_t)] = \mathbb{E}_{S_t}\big[\mathbf{1}[B \subseteq S_t]\,\mathbb{E}_q\,\delta_q(S_t)\big] = 0$. Retrieval by relevance makes S_t depend on q_t, so the logs of Section 5 need not be exogenous, and the values LEDGER reports on them describe the association between retrieval and outcome in the log. Identifiability involves $\mathbf{X}_d$ alone, so it is unaffected; only estimation is. Appending query features $\psi(q_t)$ as columns of the design removes the part of $\boldsymbol{\delta}$ in their span, and by Lemma A.3 every result on identifiability in Section 3 holds for the augmented design. Estimation needs one further condition, on adaptivity: Proposition 3 requires that the retrieved sets do not depend on past outcomes. LEDGER's rule Eq. (8) depends on the past only through $\mathbf{X}$, so it satisfies this condition.

A.4 The design of a retrieval log

The linear model. Under Assumptions 1 and 2, $r_t = v(S_t) + \varepsilon_t$ with v d-additive. By Definition A.2,

$$v(S_t) = \sum_{|B| \le d} \theta_B\, \mathbf{1}[B \subseteq S_t], \qquad \text{so} \qquad \mathbf{r} = \mathbf{X}_d\boldsymbol{\theta} + \boldsymbol{\varepsilon},$$

which is Eq. (2). The design $\mathbf{X}_d$ has $p_d = \sum_{\ell=0}^{d} \binom{n}{\ell}$ columns, one for each set of at most d memories, and its column for $\emptyset$ is $\mathbf{1}_T$. At $d = 1$, $\mathbf{X}_1 = [\mathbf{1}_T \mid \mathbf{X}]$, and its Gram matrix is Eq. (3): the number of episodes T, the retrieval frequencies $\mathbf{f}$ and the co-retrieval counts $\mathbf{X}^\top\mathbf{X}$ of Eq. (1). By Appendix A.1, these counts decide every question of estimability at $d = 1$.

Determined values. The paper uses the following definition throughout.

Definition A.4 (Determined value, fraction determined)**.** At resolution d, memory i is *identifiable*, and its value is *determined*, if φ_i is estimable in $\mathbf{r} = \mathbf{X}_d\boldsymbol{\theta} + \boldsymbol{\varepsilon}$, that is, if $\mathbf{c}^{(i)} \in \operatorname{range}\mathbf{X}_d^\top$. Otherwise its value is *undetermined*. The fraction determined is the number of identifiable memories divided by n; after merging, it is the number of determined units divided by the number of units.

At $d = 1$, $\mathbf{c}^{(i)} = (0, \mathbf{e}_i)$, so memory i is identifiable exactly when $(0, \mathbf{e}_i)$ is orthogonal to $\ker\mathbf{X}_1$. Numerically, LEDGER computes an orthonormal basis $\mathbf{U}$ of $\ker\mathbf{X}_d$ and declares memory i determined when $\|\mathbf{U}^\top\mathbf{c}^{(i)}\| \le \tau\|\mathbf{c}^{(i)}\|$ (Section 4). Unless stated otherwise, every fraction in the paper is computed at $d = 1$ with the baseline column included.

Blocks. The co-retrieval graph has one vertex per memory and an edge between m_i and m_j whenever $\operatorname{co}(m_i, m_j) > 0$.

Definition A.5 (Blocks)**.** The blocks of a log are the connected components of the co-retrieval graph restricted to the memories retrieved at least once. Block b contains n_b memories and the T_b episodes whose retrieved sets lie in it.

The members of one retrieved set are pairwise adjacent, so every episode belongs to exactly one block.

Lemma A.5 (Block rank)**.** *After permuting rows and columns, the columns of* $\mathbf{X}$ *for the retrieved memories form a block-diagonal matrix with diagonal blocks* $\mathbf{X}^{(b)} \in \{0,1\}^{T_b \times n_b}$*, and the columns for memories never retrieved are zero. Hence* $\operatorname{rank}\mathbf{X} = \sum_b \operatorname{rank}\mathbf{X}^{(b)} \le \sum_b \min(T_b, n_b)$.

The baseline column $\mathbf{1}_T$ meets every block, so $\mathbf{X}_1$ is not block-diagonal, and $\operatorname{rank}\mathbf{X}_1 \le \operatorname{rank}\mathbf{X} + 1$ by Lemma A.3. A memory that is never retrieved has a zero column, so $\mathbf{e}_i$ lies in $\ker\mathbf{X}_1$ and its value is undetermined.

Tied memories. If memories i and j are retrieved in exactly the same episodes, their columns of $\mathbf{X}$ are equal, and $\mathbf{e}_{\{i\}} - \mathbf{e}_{\{j\}} \in \ker\mathbf{X}_1$. Its inner product with $\mathbf{c}^{(i)}$ is 1, so neither value is determined, while the sum $\theta_{\{i\}} + \theta_{\{j\}}$ is orthogonal to this change. Merging (Section 4) replaces each such group by one unit valued by this sum.

The baseline test. Whether a log contains a change of the kind of $\mathbf{h}_\emptyset$ is decided by one linear system.

Definition A.6 (Baseline test)**.** The baseline test solves $\mathbf{X}\mathbf{w} = \mathbf{1}_T$ by least squares and reports the residual $\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{1}_T\|$.

A zero residual means $\mathbf{1}_T \in \operatorname{range}\mathbf{X}$. Then $(-1, \hat{\mathbf{w}}) \in \ker\mathbf{X}_1$: lowering the baseline by s and raising each $\theta_{\{i\}}$ by $s\hat{w}_i$ leaves every predicted outcome unchanged, and every memory with $\hat{w}_i \neq 0$ is undetermined. A nonzero residual means that no change of this form exists, and by Lemma A.3 it is equivalent to $\operatorname{rank}\mathbf{X}_1 = \operatorname{rank}\mathbf{X} + 1$. The test says something about retrieval only when $\operatorname{rank}\mathbf{X} < T$: when $\operatorname{rank}\mathbf{X} = T$, $\mathbf{1}_T$ lies in $\operatorname{range}\mathbf{X} = \mathbb{R}^T$ whatever the retrieval. In a top-k log, $\hat{\mathbf{w}} = \mathbf{1}_n/k$ solves the system exactly, which gives $\mathbf{h}_\emptyset$; under deterministic top-k at several sizes, nesting can make the residual zero as well, and Section 5.3 measures it.

The example of Figure 1. In the left log, each of the $\binom{4}{2} = 6$ pairs of four memories is retrieved once. Every row of $\mathbf{X}$ has two ones, so $\mathbf{X}\mathbf{w} = \mathbf{1}_6$ with $\mathbf{w} = \frac{1}{2}\mathbf{1}_4$, and $\ker\mathbf{X}_1$ is spanned by $\mathbf{h}_\emptyset = (-2, 1, 1, 1, 1)$, since $\operatorname{rank}\mathbf{X}_1 = \operatorname{rank}\mathbf{X} = 4$ for five parameters. Each $(0, \mathbf{e}_i)$ has inner product 1 with $\mathbf{h}_\emptyset$, so no memory is identifiable, while each difference $\theta_{\{i\}} - \theta_{\{j\}}$ is orthogonal to it and is determined. The right log adds one episode of size 1 and one of size 3. Their rows break $\mathbf{X}\mathbf{w} = \mathbf{1}_8$: a solution would need $w_i = \frac{1}{2}$ for every i from the pairs, and then the episode of size 1 has row sum $\frac{1}{2} \neq 1$. Hence $\operatorname{rank}\mathbf{X}_1 = 5$, the kernel is trivial, and all four memories are identifiable.

A.5 Optimal design and confidence sets

The variance of an estimate. With $\mathbf{V} = \mathbf{X}_d^\top\mathbf{X}_d + \lambda\mathbf{I}$ and noise covariance $\sigma^2\mathbf{I}$, the ridge estimate of Eq. (5) satisfies

$$\operatorname{Var}\big(\mathbf{c}^\top\hat{\boldsymbol{\theta}}\big) = \sigma^2\,\mathbf{c}^\top\mathbf{V}^{-1}\mathbf{X}_d^\top\mathbf{X}_d\mathbf{V}^{-1}\mathbf{c} = \sigma^2\big(\mathbf{c}^\top\mathbf{V}^{-1}\mathbf{c} - \lambda\,\|\mathbf{V}^{-1}\mathbf{c}\|^2\big),$$

since $\mathbf{X}_d^\top\mathbf{X}_d = \mathbf{V} - \lambda\mathbf{I}$. The first term dominates for small λ, so $\operatorname{tr}(\mathbf{V}^{-1})$ is, up to σ^2, the total variance of the coordinate estimates. A-optimal design chooses the design to minimise it (Pukelsheim, 2006). For invertible $\mathbf{V}$, the Sherman–Morrison identity

$$(\mathbf{V} + \mathbf{x}\mathbf{x}^\top)^{-1} = \mathbf{V}^{-1} - \frac{\mathbf{V}^{-1}\mathbf{x}\mathbf{x}^\top\mathbf{V}^{-1}}{1 + \mathbf{x}^\top\mathbf{V}^{-1}\mathbf{x}}$$

shows that adding an episode with row $\mathbf{x}$ reduces $\operatorname{tr}(\mathbf{V}^{-1})$ by $\|\mathbf{V}^{-1}\mathbf{x}\|^2/(1+\mathbf{x}^\top\mathbf{V}^{-1}\mathbf{x})$, and updates $\mathbf{V}^{-1}$ in $O(p_d^2)$ time.

The bound behind LEDGER's retrieval rule. For every positive definite $\mathbf{V}$ and every i,

$$(\mathbf{V}^{-1})_{ii} \;\geq\; \frac{1}{V_{ii}},$$

by the Cauchy–Schwarz inequality: $1 = (\mathbf{e}_i^\top\mathbf{V}^{1/2}\mathbf{V}^{-1/2}\mathbf{e}_i)^2 \leq (\mathbf{e}_i^\top\mathbf{V}\mathbf{e}_i)(\mathbf{e}_i^\top\mathbf{V}^{-1}\mathbf{e}_i)$. At $d = 1$, the diagonal entry of $\mathbf{V}$ for memory i is $f_i + \lambda$, so the variance of $\hat{\theta}_{\{i\}}$ is, for small λ, at least $\sigma^2/(f_i + \lambda)$. Memories retrieved rarely therefore have the largest lower bound on their variance, and LEDGER's rule Eq. (8) retrieves them first among the candidates R_t. The rule is motivated by A-optimal design and optimises no criterion: the bound is a lower bound, and the rule uses only the counts $f_i(t)$.

Confidence sets. For ridge regression with conditionally σ-sub-Gaussian noise, an adaptively chosen design and $\|\boldsymbol{\theta}\| \leq S$, Abbasi-Yadkori et al. (2011) show that, with probability at least $1 - \delta$,

$$\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_{\mathbf{V}} \leq \sigma\sqrt{2\log\Big(\frac{\det(\mathbf{V})^{1/2}\det(\lambda\mathbf{I})^{-1/2}}{\delta}\Big)} + \lambda^{1/2}S,$$

where $\|\mathbf{z}\|_{\mathbf{V}} = (\mathbf{z}^\top\mathbf{V}\mathbf{z})^{1/2}$. By the Cauchy–Schwarz inequality this bounds $|\mathbf{c}^\top(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})|$ by $\|\mathbf{c}\|_{\mathbf{V}^{-1}}$ times the right-hand side, for every $\mathbf{c}$ at once. The set remains valid when the design depends on past outcomes, and its log-determinant grows with p_d. LEDGER's interval bounds only the n functionals $\mathbf{c}^{(i)}$ and requires a design that ignores past outcomes (Proposition 3); Section 5.4 compares the two widths.

A.6 Related work

This work meets six lines of research: memory management by proxy statistics, memory value learned from outcomes, cooperative-game attribution, budgeted selection, graph-structured retrieval, and the classical theory of estimation and design. Each has produced strong systems, and this paper adopts much of their machinery. It asks a question none of them poses: which memory values a retrieval log determines.

Memory management by proxy. Persistent memory is a primary determinant of long-horizon agent performance (Pink et al., 2025; Du et al., 2025), and a body of work decides what to retain. FadeMem (Wei et al., 2026a) applies differential decay across a two-layer hierarchy, modulating an exponential forgetting rate by semantic relevance, access frequency and recency. Budget-curated forgetting (Wu et al., 2026a) formulates retention as bounded optimisation over similar statistics, and ACT-R-inspired architectures (Honda et al., 2025) import the activation equation of a model of human memory. SSGM (Lam et al., 2026) decouples memory evolution from execution with consistency checks, temporal decay and access control, and Aeon (Arslan, 2026) targets the throughput of the same operations. These systems differ in the statistic they use and share its status: it is a function of content and access, fixed before any outcome is observed, and validated by downstream accuracy. Two memories with the same content and access pattern therefore receive the same score whatever they contribute. Access frequency is the diagonal of $\mathbf{X}^\top\mathbf{X}$ in Eq. (1), so these statistics record how often a memory is used, which decides whether its contribution can be learned and does not measure it (Appendix H.3).

Memory value learned from outcomes. A second line learns a value signal from downstream reward. RMM (Tan et al., 2025) refines retrieval online by reinforcement learning from the evidence the model cites. MemRL (Zhang et al., 2026) applies reinforcement learning to episodic memory updates, with a two-phase retrieval that filters for high-utility entries. AttriMem (Li et al., 2026) uses token-level attribution to the final answer as a local reward for a memory-construction policy, and Memory-R1 (Yan et al., 2026b) and Memory-R2 (Yan et al., 2026a) train memory operations from outcome rewards, the latter with explicit credit assignment over operations. RoMeRL (Yang et al., 2026) names the memory-reward trap: irrelevant memories accrue positive credit because they are retrieved together with useful ones. These methods learn a signal and show that it helps; none characterises when the signal is recoverable. The memory-reward trap is the invisible change $\mathbf{e}_{\{i\}} - \mathbf{e}_{\{j\}}$ of tied memories (Section 3): when two memories share a retrieval history, no procedure that reads the log can separate their credit, and the answer consistent with the data is to report the pair as undetermined and merge it. When every rollout retrieves the same number of memories, the credit of every memory is undetermined (Theorem 1). Both statements constrain every method in this line, LEDGER included.

Cooperative-game attribution. The Shapley value (Shapley, 1953) is the canonical answer to what a participant contributes to a joint outcome, and Data Shapley (Ghorbani & Zou, 2019) brought it to machine learning. It has since been applied to retrieved content. Ye & Yoganarasimhan (2025) value source documents in summaries with a clustering approximation and accuracy bounds. MaxShapley (Patel et al., 2025) obtains exact values in polynomial time from a decomposable max-sum utility. Shapley context pruning (Chen et al., 2026b) and RepoShapley (Huo et al., 2026) apply the same machinery to passage reranking and to repository-level code context, and MemLens (Wei et al., 2026b) defines a Memory Shapley value and uses it to drive retention. General estimators approximate interactions of any order by sampling coalitions (Tsai et al., 2023; Fumagalli et al., 2023; Kang et al., 2025), and context attribution fits a surrogate to random ablations of the prompt (Cohen-Wang et al., 2024). We adopt the same notion of value. The difference is access. Each method above evaluates $v(S)$ on coalitions of its choosing, which requires replaying the system on subsets that never occurred. A deployed agent has only the coalitions that did occur, and LEDGER asks which values those determine. Because the Shapley value is a fixed linear functional of the Möbius dividends (Harsanyi, 1963; Grabisch et al., 2000) (Lemma A.4), the answer is exact and requires no sampling, and the error of truncating to resolution d is written in closed form (Proposition A.1).

Budgeted selection. Retention under a budget is a selection problem, and submodular maximisation supplies its standard guarantees (Nemhauser et al., 1978; Conforti & Cornuéjols, 1984; Buchbinder et al., 2014). PACMS (Ghulyani et al., 2026) applies a facility-location objective with lazy

greedy selection to pooled memory entries, turns and tool outputs. These guarantees hold for a proxy objective defined in embedding space, and coverage in that space is assumed to track usefulness. LEDGER's retention objective (Appendix F) is the estimated value itself. We do not assume that it is submodular, and we claim no guarantee that would require it.

Graph-structured retrieval and coarsening. Retrieval-augmented generation (Lewis et al., 2020) conditions generation on a retrieved set, and graphs structure what is retrieved. HippoRAG (Jiménez Gutiérrez et al., 2024) combines a knowledge graph with Personalized PageRank, G-Retriever (He et al., 2024) retrieves subgraphs as a prize-collecting Steiner tree, and RA-Graph (Jiang et al., 2024) provides a general framework for retrieval-augmented graph learning. For multi-hop reasoning, RoG (Luo et al., 2024) generates relation paths as plans, ToG-2 (Ma et al., 2025) alternates knowledge-guided and context-enhanced retrieval, StructRAG (Li et al., 2025) structures information at inference time, and HopRAG (Liu et al., 2025) traverses passages with logic-aware hops. GAM (Wu et al., 2026b) and TierMem (Zhu et al., 2026) organise agent memory hierarchically. The graphs in these systems encode relations between contents. The graph in this paper encodes usage: an edge joins two memories retrieved in the same episode, and by Eq. (1) its weighted adjacency matrix is the Gram matrix of the valuation problem. Merging is likewise distinct from graph coarsening (Joly & Keriven, 2024; Han et al., 2024), whose guarantees state that a coarsened graph approximates the original within a constant. Merging memories with identical retrieval histories keeps every determined value exactly (Theorem 2), because the only quantities it removes are the splits of each group's total, which the log never determined.

Estimation, design and bandits. Estimability of linear functionals under a rank-deficient design is classical (Rao, 1973), and so is the aliasing of effects that never vary apart (Rosenbaum & Zubizarreta, 2025). Our contribution here is the identity Eq. (1): the design matrix of memory valuation is determined by counts that memory systems already record, so these results apply to them. LEDGER's intervals are per-functional; the self-normalised confidence set of Abbasi-Yadkori et al. (2011) bounds every direction at once and remains valid under adaptive designs, at a width we compare in Section 5.4. Because only a set-level outcome is observed, the setting has the full-bandit feedback of combinatorial and top-k bandits (Chen et al., 2013; Rejwan & Mansour, 2020), and this is why the design, and not the reward model, limits what can be learned. LEDGER's retrieval rule is motivated by optimal design (Pukelsheim, 2006), with one constraint that classical design does not have: it may choose only among memories relevant to the query, which is the source of the accuracy cost measured in Section 5.3.

Slate evaluation and intervention harvesting. Two neighbours are close enough to state the difference precisely. Slate off-policy evaluation (Swaminathan et al., 2017) builds the same Gram matrix from logged set-valued actions. Its estimand is the value of a policy, for which the kernel can be projected out; ours is the value of each memory, and the kernel decides which of those the log determines. Recent work there reduces variance by learning an abstraction of the slate space (Kiyohara et al., 2024), which resembles merging; merging is forced by Theorem 2 and is exact. Intervention harvesting (Agarwal et al., 2019) recovers position bias from variation that occurs naturally across deployed rankers. LEDGER addresses the complementary case, in which that variation does not occur, and measures what it costs to create it (Section 5.3).

Evaluation protocols. Benchmarks for long-horizon memory and multi-hop retrieval, including LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2025) and HotpotQA (Yang et al., 2018), measure answer accuracy under a fixed retrieval protocol, and they measure it well. Their protocols give each question its own memories or retrieve a fixed number of items, and under either protocol the log determines no memory value (Section 5.2). Appendix E lists the fourteen benchmarks we audit and the protocol of each.

B PROOFS

This appendix proves every result of Sections 3 and 4, in the order in which they appear, together with four supporting results: the form of the Gram matrix at every resolution (Lemma B.1), a sharper form of Proposition 1 (Corollary B.1), a sufficient condition for nested sets to leave a change like $\mathbf{h}_\emptyset$ (Lemma B.2), and the monotonicity of identifiability in the resolution (Lemma B.3). Throughout,

$1 \le d \le n$, every retrieved set is non-empty, and $\mathbf{e}_B$ denotes the unit vector of $\mathbb{R}^{p_d}$ for the coordinate θ_B. We use the facts of Appendix A.1: φ_i is determined exactly when $(\mathbf{c}^{(i)})^\top \mathbf{h} = 0$ for every $\mathbf{h} \in \ker \mathbf{X}_d$ (Lemma A.1), and $\ker \mathbf{X}_d = \ker(\mathbf{X}_d^\top \mathbf{X}_d)$.

B.1 Counts decide identifiability

Lemma B.1 (Gram matrix at resolution d). *For all sets B, B' of at most d memories, $(\mathbf{X}_d^\top \mathbf{X}_d)_{BB'} = |\{t \in [T] : B \cup B' \subseteq S_t\}|$. At $d = 1$, $\mathbf{X}_1^\top \mathbf{X}_1$ is the matrix of Eq.* (3).

Proof. $(\mathbf{X}_d^\top \mathbf{X}_d)_{BB'} = \sum_t \mathbf{1}[B \subseteq S_t]\,\mathbf{1}[B' \subseteq S_t] = \sum_t \mathbf{1}[B \cup B' \subseteq S_t]$. At $d = 1$, the pairs $(\emptyset, \emptyset)$, $(\emptyset, \{i\})$ and $(\{i\}, \{j\})$ give T, f_i and $\mathrm{co}(m_i, m_j)$, with $\mathrm{co}(m_i, m_i) = f_i$. □

Since $\ker \mathbf{X}_d = \ker(\mathbf{X}_d^\top \mathbf{X}_d)$, the set of determined values depends on the log only through these counts: at $d = 1$ through T, the frequencies and the co-retrieval counts, and at resolution d through the number of episodes that retrieve each set of at most $2d$ memories.

B.2 A fixed retrieval size

Proof of the kernel identity Eq. (4). Let $|B| < d$, so that every set $B \cup \{j\}$ indexes a column of $\mathbf{X}_d$. The entry of $\mathbf{X}_d \mathbf{h}_B$ for episode t is

$$\sum_{j \notin B} \mathbf{1}[B \cup \{j\} \subseteq S_t] - (k - |B|)\,\mathbf{1}[B \subseteq S_t].$$

If $B \not\subseteq S_t$, both terms vanish. If $B \subseteq S_t$, the sum counts the members of $S_t \setminus B$, which number $|S_t| - |B| = k - |B|$, so the entry is 0. □

Proof of Theorem 1. Identifiability. Take $B = \emptyset$, which satisfies $|B| < d$ because $d \ge 1$: $\mathbf{h}_\emptyset = \sum_{j=1}^n \mathbf{e}_{\{j\}} - k\,\mathbf{e}_\emptyset \in \ker \mathbf{X}_d$. Since $c_\emptyset^{(i)} = 0$ and $c_{\{j\}}^{(i)} = \mathbf{1}[j = i]$,

$$(\mathbf{c}^{(i)})^\top \mathbf{h}_\emptyset = \sum_{j=1}^n c_{\{j\}}^{(i)} = 1 \neq 0$$

for every memory i. By Lemma A.1, no φ_i is estimable. The argument uses only that every row of $\mathbf{X}$ has k ones, so it holds for every $d \ge 1$ and every T.

Consistency. Let outcomes be binary with $\Pr(r_t = 1 \mid \mathcal{F}_{t-1}, q_t, S_t) = v_{q_t}(S_t)$, where each v_q takes values in $(0, 1)$, and let the queries and the retrieval rule be fixed. The law of the log $(q_t, S_t, r_t)_{t \le T}$ depends on the functions v_q only through their values on sets of size k, since no other set is ever retrieved. Define $u(S) = |S| - k$ and, for $s \in \mathbb{R}$, $v_q^s = v_q + s\,u$. Because each v_q takes finitely many values, all in $(0, 1)$, there is $\epsilon > 0$ with $\epsilon \le v_q(S) \le 1 - \epsilon$ for all q and S; for $|s| < \epsilon/n$, every v_q^s takes values in $(0, 1)$, so the perturbed model is valid. Moreover $v_q^s(S) = v_q(S)$ whenever $|S| = k$, so the law of the log is the same under (v_q) and (v_q^s) for every T. The dividends of u are $-k$ on $\emptyset$, 1 on each singleton and 0 otherwise, that is, u has dividend vector $\mathbf{h}_\emptyset$. Averaging over queries, $v^s = v + s\,u$ has dividends $\boldsymbol{\theta} + s\,\mathbf{h}_\emptyset$, which is d-additive whenever $\boldsymbol{\theta}$ is, and by Lemma A.4 $\varphi_i(v^s) = \varphi_i(v) + s$. Suppose an estimator $\hat{\varphi}_{i,T}$, a function of the first T episodes, converged in probability to φ_i under every valid model. Fix $0 < s < \epsilon/n$. Under (v_q) it converges to $\varphi_i(v)$, and under (v_q^s), which gives the same law for every T, it converges to $\varphi_i(v) + s$. A sequence of random variables with a fixed law for each T cannot converge in probability to two different constants, a contradiction. □

B.3 Sizes needed

Proof of Proposition 1. Let K be the set of retrieval sizes, with $|K| \le d$. For a polynomial P of degree at most d, write $P(x) = \sum_{\ell=0}^d a_\ell \binom{x}{\ell}$; the binomial polynomials $\binom{x}{0}, \ldots, \binom{x}{d}$ form a basis of

the polynomials of degree at most d, so the coefficients a_ℓ exist and are unique. Define $\mathbf{h}^P \in \mathbb{R}^{p_d}$ by $h^P_B = a_{|B|}$. For every episode t,

$$(\mathbf{X}_d \mathbf{h}^P)_t = \sum_{B \subseteq S_t,\, |B| \le d} a_{|B|} = \sum_{\ell=0}^{d} \binom{|S_t|}{\ell} a_\ell = P(|S_t|).$$

Take $P(x) = \prod_{k \in K}(x-k)$, a nonzero polynomial of degree $|K| \le d$ that vanishes at every retrieval size. Then $\mathbf{X}_d \mathbf{h}^P = \mathbf{0}$, and $\mathbf{h}^P \neq \mathbf{0}$ because $P \neq 0$ and $d \le n$, so every coefficient a_ℓ with $\ell \le d$ is carried by at least one coordinate. The construction does not depend on T. □

If every dividend were determined, $\ker \mathbf{X}_d$ would be trivial, so the proposition implies that a log with at most d distinct sizes leaves some dividend undetermined. The same construction shows more: under a mild condition on the sizes, it leaves every memory value undetermined.

Corollary B.1 (Few sizes leave every value undetermined)**.** *If the retrieval sizes take at most d distinct values and every size is smaller than $n/2$, no memory is identifiable at resolution d.*

Proof. For the vector $\mathbf{h}^P$ of the previous proof,

$$(\mathbf{c}^{(i)})^\top \mathbf{h}^P = \sum_{\ell=1}^{d} \binom{n-1}{\ell-1} \frac{a_\ell}{\ell} = \frac{1}{n} \sum_{\ell=1}^{d} \binom{n}{\ell} a_\ell = \frac{P(n) - P(0)}{n},$$

using $\binom{n-1}{\ell-1}/\ell = \binom{n}{\ell}/n$ and $a_0 = P(0)$. With $P(x) = \prod_{k \in K}(x-k)$ and $1 \le k < n/2$ for every $k \in K$, we have $n - k > k$, so $|P(n)| = \prod_k (n-k) > \prod_k k = |P(0)|$, hence $P(n) \neq P(0)$, and the inner product is nonzero for every i. □

At $d = 1$ with one size k, $P(x) = x - k$ gives $\mathbf{h}^P = \mathbf{h}_\emptyset$ and $(P(n) - P(0))/n = 1$, recovering Theorem 1 without the condition $k < n/2$. The condition cannot be dropped in general: with $n = 6$, $d = 2$ and every set of sizes 2 and 4 retrieved, $P(6) = P(0) = 8$, and every memory is identifiable although $\ker \mathbf{X}_2 \neq \{\mathbf{0}\}$. Conversely, $d + 1$ distinct sizes are necessary for every dividend to be determined, and they are not sufficient: which values are determined still depends on which sets are retrieved.

B.4 Nested sets

Under deterministic top-k, each query q has a fixed ranking of the memories, and every set retrieved for q is a prefix of it. Let k_1 be the smallest retrieval size, let A_q be the k_1 memories ranked highest for q, and let $C = \bigcup_q A_q$.

Lemma B.2 (Nesting leaves a change like $\mathbf{h}_\emptyset$)**.** *Suppose every set S retrieved for any query q satisfies $S \cap C = A_q$. Then $\mathbf{w} = \mathbf{1}_C/k_1$ solves $\mathbf{X}\mathbf{w} = \mathbf{1}_T$, the vector $(-1, \mathbf{w})$ lies in $\ker \mathbf{X}_1$, and no memory in C is identifiable at $d = 1$.*

Proof. Every retrieved set contains A_q, because the sets for q are prefixes of one ranking of length at least k_1. By hypothesis it contains no other member of C, so $(\mathbf{X}\mathbf{w})_t = |S_t \cap C|/k_1 = |A_{q_t}|/k_1 = 1$. Then $\mathbf{X}_1(-1, \mathbf{w}) = -\mathbf{1}_T + \mathbf{X}\mathbf{w} = \mathbf{0}$, and for $i \in C$ the inner product of $(0, \mathbf{e}_i)$ with $(-1, \mathbf{w})$ is $1/k_1 \neq 0$. □

The hypothesis holds whenever the queries have private stores, and in a shared store it holds when no set retrieved for one query contains a memory ranked in the top k_1 for another. The lemma does not require a single retrieval size, so varying the size does not remove this change. The baseline test (Definition A.6) detects the resulting change whether or not the hypothesis holds exactly: under deterministic top-k on pooled HotpotQA its residual is 3.1×10^{-15}, zero up to rounding (Section 5.3).

B.5 PRIVATE MEMORIES

Proof of Proposition 2. Order the memories by query and the episodes by query. Because query b retrieves only from its own n_b memories, $\mathbf{X}$ is block-diagonal with blocks $\mathbf{X}^{(b)} \in \{0,1\}^{T_b \times n_b}$ (Lemma A.5). For any $\mathbf{g} \in \ker \mathbf{X}^{(b)}$, the vector that equals $\mathbf{g}$ on the memories of b and 0 on the baseline and on every other memory lies in $\ker \mathbf{X}_1$. If memory i of query b is identifiable, $(0, \mathbf{e}_i)$ is orthogonal to all these vectors, so $\mathbf{e}_i$, restricted to the memories of b, lies in $(\ker \mathbf{X}^{(b)})^\perp = \operatorname{range} \mathbf{X}^{(b)\top}$. The identifiable memories of b thus give linearly independent vectors in a space of dimension $\operatorname{rank} \mathbf{X}^{(b)} \le \min(T_b, n_b)$, so at most T_b of them are identifiable and at least $\max(n_b - T_b, 0)$ are not. Summing over queries gives the bound. □

B.6 RESOLUTION

Lemma B.3 (Undetermined values stay undetermined)**.** *If memory i is not identifiable at resolution d, it is not identifiable at any resolution $d' > d$.*

Proof. It suffices to take $d' = d + 1 \le n$. The columns of $\mathbf{X}_{d+1}$ are those of $\mathbf{X}_d$ followed by the columns for sets of size $d+1$. Let $\mathbf{h} \in \ker \mathbf{X}_d$ with $(\mathbf{c}^{(i)})^\top \mathbf{h} \neq 0$, and let $\tilde{\mathbf{h}}$ extend $\mathbf{h}$ by zeros. Then $\mathbf{X}_{d+1}\tilde{\mathbf{h}} = \mathbf{X}_d \mathbf{h} = \mathbf{0}$. The coefficient vector of φ_i at resolution $d+1$ agrees with $\mathbf{c}^{(i)}$ on every set of size at most d, since both equal $\mathbf{1}[i \in B]/|B|$, so its inner product with $\tilde{\mathbf{h}}$ is $(\mathbf{c}^{(i)})^\top \mathbf{h} \neq 0$. □

B.7 INTERVALS

We state the conditions of Proposition 3 formally. Let $\mathcal{S} = \sigma(S_1, \dots, S_T)$. The proposition assumes that for every t and every $s \in \mathbb{R}$,

$$\mathbb{E}\big[\varepsilon_t \mid \mathcal{S}, \varepsilon_{1:t-1}\big] = 0, \qquad \mathbb{E}\big[e^{s\varepsilon_t} \mid \mathcal{S}, \varepsilon_{1:t-1}\big] \le e^{s^2\sigma^2/2}. \tag{11}$$

Conditioning on all sets, including later ones, is what requires the sets not to depend on past outcomes: if S_{t+1} depended on r_t, it would carry information about ε_t and Eq. (11) would fail in general. The mean-zero part of Eq. (11) also requires the query-specific part $\delta_{q_t}(S_t)$ of the noise to average out given the sets, which is the exogeneity of Appendix A.3.

Proof of Proposition 3. With $\mathbf{r} = \mathbf{X}_d\boldsymbol{\theta} + \boldsymbol{\varepsilon}$ and $\mathbf{X}_d^\top \mathbf{X}_d = \mathbf{V} - \lambda \mathbf{I}$,

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta} = \mathbf{V}^{-1}(\mathbf{V} - \lambda \mathbf{I})\boldsymbol{\theta} + \mathbf{V}^{-1}\mathbf{X}_d^\top \boldsymbol{\varepsilon} - \boldsymbol{\theta} = -\lambda \mathbf{V}^{-1}\boldsymbol{\theta} + \mathbf{V}^{-1}\mathbf{X}_d^\top \boldsymbol{\varepsilon}.$$

Fix $\mathbf{c} = \mathbf{c}^{(i)}$ and let $\mathbf{a} = \mathbf{X}_d \mathbf{V}^{-1}\mathbf{c}$, so that $\hat{\varphi}_i - \varphi_i = -\lambda\, \mathbf{c}^\top \mathbf{V}^{-1}\boldsymbol{\theta} + \mathbf{a}^\top \boldsymbol{\varepsilon}$. The vector $\mathbf{a}$ is a function of the sets, hence $\mathcal{S}$-measurable. Conditioning successively on $\mathcal{S}$ and $\varepsilon_{1:T-1}, \dots, \varepsilon_{1:0}$ and applying Eq. (11) at each step gives, for every $s \in \mathbb{R}$,

$$\mathbb{E}\big[e^{s\,\mathbf{a}^\top \boldsymbol{\varepsilon}} \mid \mathcal{S}\big] \le \prod_{t=1}^{T} e^{s^2 a_t^2 \sigma^2/2} = e^{s^2\sigma^2\|\mathbf{a}\|^2/2}.$$

The Chernoff bound then gives, with $u = \sigma\|\mathbf{a}\|\sqrt{2\log(2n/\delta)}$,

$$\Pr\big(|\mathbf{a}^\top \boldsymbol{\varepsilon}| > u \mid \mathcal{S}\big) \le 2\exp\Big(-\frac{u^2}{2\sigma^2\|\mathbf{a}\|^2}\Big) = \frac{\delta}{n}.$$

On the complementary event, $|\hat{\varphi}_i - \varphi_i| \le u + \lambda|\mathbf{c}^\top \mathbf{V}^{-1}\boldsymbol{\theta}|$, which is Eq. (6). A union bound over the n memories gives the simultaneous statement with probability at least $1 - \delta$. □

The proof does not use that memory i is determined; the bias term shows why the proposition is stated for determined memories. Write $\mathbf{c} = \mathbf{c}_1 + \mathbf{c}_2$ with $\mathbf{c}_1 \in \operatorname{range} \mathbf{X}_d^\top$ and $\mathbf{c}_2 \in \ker \mathbf{X}_d$. Since $\mathbf{V}\mathbf{c}_2 = \lambda \mathbf{c}_2$, $\lambda\, \mathbf{c}^\top \mathbf{V}^{-1}\boldsymbol{\theta} = \lambda\, \mathbf{c}_1^\top \mathbf{V}^{-1}\boldsymbol{\theta} + \mathbf{c}_2^\top \boldsymbol{\theta}$. For a determined memory, $\mathbf{c}_2 = \mathbf{0}$, and because $\mathbf{V}^{-1}\mathbf{c}_1$ lies in the row space, where the eigenvalues of $\mathbf{V}^{-1}$ are at most $1/(\sigma_{\min}^+(\mathbf{X}_d)^2 + \lambda)$,

$$\lambda\, |\mathbf{c}^\top \mathbf{V}^{-1}\boldsymbol{\theta}| \le \frac{\lambda\, \|\mathbf{c}\|\, \|\boldsymbol{\theta}\|}{\sigma_{\min}^+(\mathbf{X}_d)^2 + \lambda} \;\longrightarrow\; 0 \quad (\lambda \to 0).$$

For an undetermined memory, the bias contains $\mathbf{c}_2^\top \boldsymbol{\theta}$, which does not shrink with λ or with T; this is the undercoverage observed on undetermined values in Section 5.4. The interval Eq. (5) replaces σ by $\hat{\sigma}$ and $\sqrt{2\log(2n/\delta)}$ by the Gaussian quantile $z_{1-\delta/(2n)}$, which is smaller, and omits the bias term; its coverage is therefore measured empirically (Section 5.4).

B.8 MERGING

Let the groups $g_1, \dots, g_m$ partition $[n]$, and let $\mathbf{G} \in \{0,1\}^{n \times m}$ have $G_{ig} = \mathbf{1}[i \in g]$. Write $\bar{\mathbf{G}} = \operatorname{diag}(1, \mathbf{G}) \in \{0,1\}^{(1+n)\times(1+m)}$ for its extension by the baseline coordinate. Its columns have disjoint non-empty supports, so $\bar{\mathbf{G}}$ has full column rank and $\bar{\mathbf{G}}^\top$ is surjective. For $\boldsymbol{\theta} = (\theta_\emptyset, \theta_{\{1\}}, \dots, \theta_{\{n\}})$, the vector $\bar{\boldsymbol{\theta}} = \bar{\mathbf{G}}^\top \boldsymbol{\theta} = (\theta_\emptyset, s_{g_1}(\boldsymbol{\theta}), \dots, s_{g_m}(\boldsymbol{\theta}))$ collects the baseline and the group values; Theorem 3 writes $\mathbf{s}(\boldsymbol{\theta})$ for it.

Proof of Theorem 2. When each group consists of memories with identical columns, the column of $\mathbf{X}$ for memory i equals the column of $\bar{\mathbf{X}}$ for its group, so $\mathbf{X} = \bar{\mathbf{X}}\mathbf{G}^\top$ and $\mathbf{X}_1 = \bar{\mathbf{X}}_1 \bar{\mathbf{G}}^\top$. Because $\bar{\mathbf{G}}^\top$ is surjective, $\operatorname{range} \mathbf{X}_1 = \bar{\mathbf{X}}_1 \operatorname{range} \bar{\mathbf{G}}^\top = \operatorname{range} \bar{\mathbf{X}}_1$, and the least-squares fitted outcomes, the projection of $\mathbf{r}$ onto this common range, agree. For the row spaces, $\operatorname{range} \mathbf{X}_1^\top = \bar{\mathbf{G}} \operatorname{range} \bar{\mathbf{X}}_1^\top$. Hence $\mathbf{c} \in \operatorname{range} \mathbf{X}_1^\top$ exactly when $\mathbf{c} = \bar{\mathbf{G}}\bar{\mathbf{c}}$ for some $\bar{\mathbf{c}} \in \operatorname{range} \bar{\mathbf{X}}_1^\top$. The condition $\mathbf{c} = \bar{\mathbf{G}}\bar{\mathbf{c}}$ says that $c_{\{i\}} = \bar{c}_{g(i)}$ for every i and $c_\emptyset = \bar{c}_\emptyset$, that is, $\mathbf{c}$ takes one value on each group; $\bar{\mathbf{c}}$ is then unique because $\bar{\mathbf{G}}$ has full column rank. In that case $\mathbf{c}^\top \boldsymbol{\theta} = \bar{\mathbf{c}}^\top \bar{\mathbf{G}}^\top \boldsymbol{\theta} = \bar{\mathbf{c}}^\top \bar{\boldsymbol{\theta}}$, the corresponding combination of group values. □

In particular, for a memory i in a group with at least two members, $(0, \mathbf{e}_i)$ does not take one value on that group, so φ_i is undetermined, while the group value $s_{g(i)}(\boldsymbol{\theta})$ is determined exactly when $\bar{\mathbf{X}}_1$ determines it.

For an arbitrary grouping, the merged column of group g is $\bar{\mathbf{x}}_g = \max_{i \in g} \mathbf{x}_i$ entrywise, where $\mathbf{x}_i$ is the column of $\mathbf{X}$ for memory i. Let $\boldsymbol{\mu} = \mathbf{X}_1 \boldsymbol{\theta}$ be the vector of expected outcomes. The value that the merged log determines for a combination $\bar{\mathbf{c}}^\top \bar{\boldsymbol{\theta}}$ with $\bar{\mathbf{c}} \in \operatorname{range} \bar{\mathbf{X}}_1^\top$ is $\bar{\mathbf{c}}^\top \bar{\mathbf{X}}_1^\dagger \boldsymbol{\mu}$, the common value of $\bar{\mathbf{c}}^\top \boldsymbol{\beta}$ over all least-squares solutions $\boldsymbol{\beta}$ of $\bar{\mathbf{X}}_1 \boldsymbol{\beta} \approx \boldsymbol{\mu}$ (Lemma A.1); it is the limit of the merged estimate as the noise vanishes.

Proof of Theorem 3. The bound. Since $\bar{\mathbf{X}}_1 \bar{\boldsymbol{\theta}} = \theta_\emptyset \mathbf{1}_T + \sum_g s_g(\boldsymbol{\theta}) \bar{\mathbf{x}}_g = \theta_\emptyset \mathbf{1}_T + \sum_i \theta_{\{i\}} \bar{\mathbf{x}}_{g(i)}$,

$$\boldsymbol{\mu} = \bar{\mathbf{X}}_1 \bar{\boldsymbol{\theta}} + \mathbf{e}, \qquad \mathbf{e} = \sum_{i=1}^{n} \theta_{\{i\}} \big(\mathbf{x}_i - \bar{\mathbf{x}}_{g(i)} \big).$$

For $\bar{\mathbf{c}} \in \operatorname{range} \bar{\mathbf{X}}_1^\top$, $\bar{\mathbf{c}}^\top \bar{\mathbf{X}}_1^\dagger \bar{\mathbf{X}}_1 = \bar{\mathbf{c}}^\top$, because $\bar{\mathbf{X}}_1^\dagger \bar{\mathbf{X}}_1$ is the orthogonal projection onto the row space. Hence

$$\bar{\mathbf{c}}^\top \bar{\mathbf{X}}_1^\dagger \boldsymbol{\mu} - \bar{\mathbf{c}}^\top \bar{\boldsymbol{\theta}} = \bar{\mathbf{c}}^\top \bar{\mathbf{X}}_1^\dagger \mathbf{e}, \qquad \big| \bar{\mathbf{c}}^\top \bar{\mathbf{X}}_1^\dagger \mathbf{e} \big| \le \|\bar{\mathbf{c}}\| \, \|\bar{\mathbf{X}}_1^\dagger\| \, \|\mathbf{e}\| = \frac{\|\bar{\mathbf{c}}\| \, \|\mathbf{e}\|}{\sigma^+_{\min}(\bar{\mathbf{X}}_1)}.$$

Each vector $\mathbf{x}_i - \bar{\mathbf{x}}_{g(i)}$ has entries in $\{-1, 0\}$, with -1 exactly in the episodes where group $g(i)$ was retrieved and memory i was not; there are at most $D_{g(i)}$ of them, so $\|\mathbf{x}_i - \bar{\mathbf{x}}_{g(i)}\| \le \sqrt{D_{g(i)}}$. The triangle inequality gives $\|\mathbf{e}\| \le \sum_i |\theta_{\{i\}}| \sqrt{D_{g(i)}}$, which proves Eq. (7). When every $D_g = 0$, $\mathbf{e} = \mathbf{0}$ and the merged log determines every $\bar{\mathbf{c}}^\top \bar{\boldsymbol{\theta}}$ exactly, as in Theorem 2.

Sharpness. Fix integers $1 \le a < T$. Take two memories forming one group, where memory 2 is retrieved in episodes $1, \dots, a$ and memory 1 is never retrieved, and let $\boldsymbol{\theta} = (0, \theta, 0)$ with $\theta \neq 0$. Then $D_g = a$, $\bar{\mathbf{X}}_1 = [\mathbf{1}_T \mid \mathbf{1}_{[a]}]$ has full column rank, and $\boldsymbol{\mu} = \mathbf{X}_1 \boldsymbol{\theta} = \mathbf{0}$, so for $\bar{\mathbf{c}} = (0, 1)$ the merged log determines the value 0 for the group value $s_g = \theta$: the error is $|\theta|$. The bound Eq. (7) equals $|\theta| \sqrt{a} / \sigma^+_{\min}(\bar{\mathbf{X}}_1)$, where

$$\sigma^+_{\min}(\bar{\mathbf{X}}_1)^2 = \lambda_{\min} \begin{bmatrix} T & a \\ a & a \end{bmatrix} = \frac{T + a - \sqrt{(T-a)^2 + 4a^2}}{2} \;\longrightarrow\; a \quad (T \to \infty).$$

The ratio of the error to the bound therefore tends to 1. Consider any bound of the form $\|\bar{\mathbf{c}}\| \, \gamma(\sigma^+_{\min}) \sum_i |\theta_{\{i\}}| \, \rho(D_{g(i)})$. With $\gamma(\sigma) = 1/\sigma$, the family forces $\rho(a) \ge \sigma^+_{\min}(\bar{\mathbf{X}}_1) \to \sqrt{a}$

for every integer a, so $\rho(D) \geq \sqrt{D}$. With $\rho(D) = \sqrt{D}$, it forces $\sigma^+_{\min}\,\gamma(\sigma^+_{\min}) \geq \sigma^+_{\min}/\sqrt{a}$, and the right-hand side tends to 1, so no bound of this form improves $1/\sigma^+_{\min}$ by a constant factor below one on these logs. Neither factor can be improved while the other is kept. □

C ALGORITHMS

This appendix gives LEDGER's three actions as algorithms, in the order of Section 4: the test and the estimates (Algorithm 1), merging (Algorithm 2) and retrieval (Algorithm 3), followed by the baseline test and the cost of each step. The released code implements them as stated; its names for the parameters are given where they differ from the paper's.

C.1 THE TEST AND THE ESTIMATES

Algorithm 1 decides, for each memory, whether the log determines its value, and estimates only the values it determines. The test comes first, because the ridge solution returns a number for every memory (Lemma A.2). It uses the kernel of the Gram matrix, which equals $\ker \mathbf{X}_d$ (Appendix A.1) and has p_d rows whatever T is.

Algorithm 1 Test and estimate

Require: retrieved sets $S_1, \ldots, S_T$; outcomes $\mathbf{r}$; resolution d; ridge parameter λ; tolerance τ; level δ

Ensure: for each memory i: an estimate with an interval, or *undetermined*

build $\mathbf{X}_d$ with $(\mathbf{X}_d)_{tB} = \mathbf{1}[B \subseteq S_t]$ for $|B| \leq d$
$\mathbf{M} \leftarrow \mathbf{X}_d^\top \mathbf{X}_d$ ▷ counts only, Lemma B.1
$\mathbf{U} \leftarrow$ orthonormal eigenvectors of $\mathbf{M}$ with eigenvalue at most $\epsilon_{\text{rank}}\,\|\mathbf{M}\|$ ▷ basis of $\ker \mathbf{X}_d$
$\mathbf{V} \leftarrow \mathbf{M} + \lambda \mathbf{I}$; $\hat{\boldsymbol{\theta}} \leftarrow \mathbf{V}^{-1}\mathbf{X}_d^\top \mathbf{r}$
$\hat{\sigma} \leftarrow \|\mathbf{r} - \mathbf{X}_d\hat{\boldsymbol{\theta}}\| / \sqrt{T - \operatorname{tr}(\mathbf{V}^{-1}\mathbf{M})}$
for $i = 1, \ldots, n$ **do**
  $c_B^{(i)} \leftarrow \mathbf{1}[i \in B]/|B|$ for $|B| \leq d$
  if $\|\mathbf{U}^\top \mathbf{c}^{(i)}\| \leq \tau\,\|\mathbf{c}^{(i)}\|$ **then**
    $\hat{\varphi}_i \leftarrow (\mathbf{c}^{(i)})^\top \hat{\boldsymbol{\theta}}$; $w_i \leftarrow z_{1-\delta/(2n)}\,\hat{\sigma}\,\|\mathbf{X}_d\mathbf{V}^{-1}\mathbf{c}^{(i)}\|$
    report $\hat{\varphi}_i \pm w_i$
  else
    report *undetermined*
  end if
end for

In all experiments, $\lambda = 10^{-2}$, $\tau = 10^{-8}$ and $\delta = 0.05$. The basis $\mathbf{U}$ keeps the eigenvectors of $\mathbf{M}$ with eigenvalue at most $p_d\,\epsilon_{\text{mach}}\max(\lambda_{\max}(\mathbf{M}), 1)$, and $\hat{\sigma}$ uses the effective degrees of freedom $\operatorname{tr}(\mathbf{V}^{-1}\mathbf{M})$. For $p_d > 4{,}000$, the test solves one least squares problem per memory by LSMR instead of forming $\mathbf{U}$. The width uses $\|\mathbf{X}_d\mathbf{V}^{-1}\mathbf{c}\|^2 = \mathbf{c}^\top\mathbf{V}^{-1}\mathbf{M}\mathbf{V}^{-1}\mathbf{c}$, so it needs $\mathbf{M}$ and not $\mathbf{X}_d$. At $d = 1$, $\mathbf{c}^{(i)} = (0, \mathbf{e}_i)$, so the test reads the i-th row of $\mathbf{U}$ and the estimate is $\hat{\theta}_{\{i\}}$.

C.2 MERGING

Algorithm 2 groups memories with identical columns of $\mathbf{X}$ and builds the merged matrix of Theorem 2. Hashing makes it linear in the size of the log.

The memories never retrieved share the empty history and form one group with a zero column, which stays undetermined. LEDGER then applies Algorithm 1 to $\bar{\mathbf{X}}$, with one unit per group; by Theorem 2, every value determined before merging is determined after it.

Merging similar histories. Theorem 3 bounds the error of merging memories whose histories differ. Theorem 3 bounds the error of merging memories whose histories differ. LEDGER finds candidate pairs of units by locality-sensitive hashing of their columns and keeps those whose columns differ in at most $\delta_{\max} = 8$ episodes. It accepts a candidate unconditionally when the merged unit's total

Algorithm 2 Merge memories with identical histories

Require: the matrix $\mathbf{X} \in \{0,1\}^{T \times n}$
Ensure: groups $g_1, \dots, g_m$ and the merged matrix $\bar{\mathbf{X}} \in \{0,1\}^{T \times m}$
$H \leftarrow$ empty hash table from column keys to groups
for $i = 1, \dots, n$ **do**
  $\kappa \leftarrow$ the set $\{t : \mathbf{X}_{ti} = 1\}$, as a sorted tuple ▷ the retrieval history of m_i
  append i to the group $H[\kappa]$, creating it if absent
end for
$g_1, \dots, g_m \leftarrow$ the groups of H
$\bar{\mathbf{X}}_{\cdot g} \leftarrow$ the column of any member of g, for each group g
return $g_1, \dots, g_m$ and $\bar{\mathbf{X}}$

is undetermined before the merge and determined after it, and otherwise when the bound Eq. (7), with each $|\theta_{\{i\}}|$ replaced by $|\hat{\theta}_{\{i\}}| + 1.96\,\mathrm{se}_i$, gives a positive gain in determined values. On the pooled HotpotQA log of Section 5.4, it accepts two merges, both of the first kind.

C.3 Retrieval

Algorithm 3 implements the rule Eq. (8). It keeps one counter per memory, and its only state across episodes is the vector of counts $\mathbf{f}(t)$.

Algorithm 3 LEDGER's retrieval (one episode)

Require: query q_t; similarities $\mathrm{sim}(q_t, m_i)$ for all i; counts $\mathbf{f}(t)$; sizes $\mathcal{K}$ with at least $d+1$ values; window M; share ρ
Ensure: retrieved set S_t and updated counts
draw k_t from $\mathcal{K}$
$R_t \leftarrow$ the M memories most similar to q_t
$\ell_t \leftarrow \max(1, \mathrm{round}(\rho k_t))$
$L_t \leftarrow$ the ℓ_t members of R_t with the smallest $f_i(t)$
$H_t \leftarrow$ the $k_t - \ell_t$ members of R_t most similar to q_t
$S_t \leftarrow L_t \cup H_t$ ▷ $|S_t| < k_t$ when L_t and H_t overlap
$f_i(t+1) \leftarrow f_i(t) + \mathbf{1}[i \in S_t]$ for every i
return S_t

In the experiments, $\mathcal{K} = \{3, 5, 8\}$, $M = 20$ and $\rho = 0.4$, so $\ell_t = 1, 2, 3$ at $k_t = 3, 5, 8$; the code calls the rule `aopt0.4` and the window `M_CAND`. The size k_t is drawn uniformly from $\mathcal{K}$ by a generator seeded with the seed, the question and the pass, shared by every rule compared in Section 5.3, so the rules differ only in which memories they retrieve. Ties in $f_i(t)$ are broken in a fixed order that the sort determines. The rule reads the past only through $\mathbf{f}(t)$, a function of $\mathbf{X}$, so it satisfies the condition of Proposition 3. Because the counts change from pass to pass, one query retrieves different sets over time, which breaks the nesting of Lemma B.2.

C.4 The baseline test and costs

The baseline test (Definition A.6) solves $\mathbf{X}\mathbf{w} = \mathbf{1}_T$ by least squares and reports the residual $\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{1}_T\|$. A residual at the level of rounding, such as 3.1×10^{-15} on pooled HotpotQA under deterministic top-k, means that $\mathbf{1}_T$ lies in $\mathrm{range}\,\mathbf{X}$ and the log contains a change like $\mathbf{h}_\emptyset$.

Table 3 gives the cost of each step, with $\bar{k} = \frac{1}{T}\sum_t |S_t|$ and $p_d = \sum_{\ell \le d} \binom{n}{\ell}$. Building $\mathbf{M}$ touches each pair of memories in each retrieved set once, which is why its cost depends on $\bar{k}$ and not on n.

The cubic step dominates. On pooled HotpotQA, with $n = 2{,}971$, it is one eigendecomposition of a $2{,}972 \times 2{,}972$ matrix. Merging reads each retrieval once, since the history of m_i is the list of episodes that retrieved it; this is the $O(nT)$ bound of Section 4 for a dense representation of $\mathbf{X}$, and $O(T\bar{k} + n)$ for the sparse one.

Table 3: Cost of each step at $d = 1$, where $p_1 = n + 1$.

Step	Time	Memory
Gram matrix $\mathbf{M}$	$O(T\bar{k}^2 + n^2)$	$O(n^2)$
Kernel basis $\mathbf{U}$ and $\mathbf{V}^{-1}$	$O(n^3)$	$O(n^2)$
Test for all memories	$O(n \cdot \dim\ker \mathbf{X}_1)$	$O(n^2)$
Estimates and intervals	$O(n^3)$, once	$O(n^2)$
Merging (Algorithm 2)	$O(T\bar{k} + n)$ expected	$O(T\bar{k} + n)$
Retrieval, one episode	$O(n + M \log M)$	$O(n)$
Rank-one update of $\mathbf{V}^{-1}$	$O(n^2)$	$O(n^2)$

D Experimental Setup

This appendix specifies every experiment of Section 5: the corpora and stores (Appendix D.1), retrieval and reading (Appendix D.2), the retrieval rules (Appendix D.3), the synthetic logs (Appendix D.4), and the statistics (Appendix D.5). Table 4 maps each number in the main text to the run that produces it. The audit of fourteen benchmarks has its own protocol, given in Appendix E.

D.1 Corpora and stores

The retrieval experiments use three corpora with different structure.

HotpotQA. Each question of HotpotQA (Yang et al., 2018) comes with ten paragraphs, two of which support the answer. We use the paragraphs as memories and pool the paragraphs of many questions into one shared store, so that a paragraph of one question can be retrieved for another. A few validation items ship fewer than ten paragraphs, and we do not de-duplicate, so 300 questions give $n = 2{,}971$ memories. The gold memories of a question are its supporting paragraphs.

LongMemEval. Each question of LongMemEval (Wu et al., 2025) comes with a chat history. We use its turns as memories and pool the histories of the questions into one shared store. We use the oracle split. The gold memories of a question are its turns flagged `has answer`; the accuracy comparison uses the 479 of 500 questions that have at least one such turn.

LoCoMo. LoCoMo (Maharana et al., 2024) contains long conversations, each with its own questions. We use the turns of a conversation as memories, and all questions of a conversation share its store; retrieval for a question is restricted to its conversation. The gold memories of a question are the turns cited as its evidence. 9 of the 1,542 questions cite no evidence and are excluded from gold recall.

D.2 Retrieval and reading

Every memory and every question is embedded with the sentence encoder `all-MiniLM-L6 v2` (Reimers & Gurevych, 2019), normalised to unit length, so similarity is the cosine of the embeddings. All rules rank memories by this similarity; they differ in which memories of the ranking they retrieve (Appendix D.3). The reader is `gemini-2.5-flash-lite` at temperature 0, prompted with the question and the retrieved memories in retrieval order, up to 24,000 characters. An answer that matches a gold answer exactly after normalization is correct; otherwise the same model, with a fixed judge prompt, decides. Both prompts are released with the code. Two further readers, `gemini-3.6-flash` and `gemini-3.7-flash`, are used only to measure the cost of pooling a store on LongMemEval (Appendix F.4). Hosted models cannot be pinned, so accuracies are reproducible in design and not bit for bit; the retrieval logs and every identifiability number regenerate byte for byte from the released inputs.

D.3 RETRIEVAL RULES

Four rules are compared in Section 5.3. All four draw the size k_t of each retrieval from $\mathcal{K} = \{3, 5, 8\}$ with the same seeded function of the question and the pass, so they differ only in which memories they retrieve.

- *Deterministic top-k* retrieves the k_t memories most similar to the query.
- *LEDGER* applies the rule Eq. (8) with $M = 20$ and $\rho = 0.4$ (Algorithm 3); the code calls it `aopt0.4`.
- *ε-greedy* replaces, with probability ε, the top-k_t set by a uniformly random k_t-subset of the whole store; we use $\varepsilon \in \{0.1, 0.3\}$.
- *Score perturbation* adds independent Gumbel noise of scale β to each similarity and retrieves the top k_t of the perturbed scores; we use $\beta \in \{0.05, 0.10\}$.

LEDGER's sets fall below the drawn size when L_t and H_t overlap. On pooled HotpotQA this happens in 1.0%, 3.3% and 15.9% of episodes at $k_t = 3$, 5 and 8, for a mean size of 5.25 against 5.33 for the other rules, whose sets always have size k_t. The realized sizes take at least three distinct values, so Proposition 1 is satisfied at $d = 1$ and $d = 2$. The LoCoMo merging run of Section 5.4 uses score perturbation with $\beta = 0.08$.

D.4 SYNTHETIC LOGS

Coverage requires the true values, so the intervals are tested on synthetic logs drawn from the model Eq. (2). A setting fixes the number of memories $n \in \{10, 30\}$, the retrieval size $k \in \{3, 5\}$, the ratio $T/n \in \{2, 10, 50\}$ and the ridge parameter $\lambda \in \{10^{-4}, 10^{-2}, 10^{-1}\}$, which gives 36 settings at each resolution $d \in \{1, 2\}$. For each setting we draw 20 random designs and, for each design, 100 noise vectors, and we record whether every interval that Algorithm 1 reports contains its true value. Joint coverage is the fraction of the $2{,}000$ draws in which all reported intervals do. Each design draws, for every episode, a size uniformly from $\{1, \dots, k\}$ and a uniformly random set of that size, so k is the largest size. The dividends are independent standard normal, and the noise is standard normal with $\sigma = 1$, which the interval is given. At $d = 1$ every design has full rank; at $d = 2$, 24 of the 36 settings give rank-deficient designs. The restriction of coverage to reported values (Section 5.4) is a post hoc analysis of the same draws; Appendix G records it as such.

The baselines of the width comparison are the pointwise Gaussian interval, which uses the true σ and no multiplicity correction, and the self-normalised confidence set of Abbasi-Yadkori et al. (2011), evaluated at $\lambda = 1$ on a separate grid of n, k and T (Appendix F).

D.5 STATISTICS

Every accuracy comparison is paired: two rules answer the same questions, and we report the mean difference in accuracy with a 95% interval computed by $10{,}000$ paired bootstrap resamples over questions (percentile interval) and a two-sided paired permutation test with $10{,}000$ sign flips. The four comparisons against the random rules share a family and are Holm-corrected. Fractions determined on pooled HotpotQA are averaged over five seeds (0 to 4), with a range across seeds reported in Appendix F. Every claim with a decision rule was pre-registered before its measurement; Appendix G lists the pre-registered claims, the three that were withdrawn, and the one post hoc analysis.

D.6 RUNS BEHIND THE MAIN TEXT

Compute. All identifiability computations, merging and synthetic experiments run on a CPU; the largest step is one eigendecomposition of a $2{,}972 \times 2{,}972$ matrix per log. Reader and judge calls go to hosted APIs; the number of calls per experiment is not logged per run; the per-row records account for at least $16{,}742$ reader calls.

Table 4: The run behind each result in Section 5. Artifacts are named as in the released code.

Result	Design	Artifact
Audit, 55 configurations (Table 2)	fourteen benchmarks, one pass, their own retrieval	`audit_stageB.csv`
Fraction determined, four rules (Figure 3a)	pooled HotpotQA, 300 questions × 24 passes, $n = 2{,}971$, $T = 7{,}200$, seeds 0–4	`e1_` `identifiability.csv`
Baseline-test residuals	same log as above	`e1_ident_` `both_bases.csv`
Accuracy, LEDGER against top-k	pooled HotpotQA, 1,600 held-out questions, store of 15,900 memories	`e1_primary.csv`
Accuracy, random rules	pooled HotpotQA, 300 questions, one pass, Holm-corrected	`e1_frontier_` `table.csv`
Accuracy on LoCoMo and LongMemEval (Figure 3b)	1,542 and 479 questions, deterministic against `aopt0.4`	`e1_replicate_` `summary.csv`
Mechanism and oracle (Figure 3c)	pooled HotpotQA, split by gold recall of top-k	`e1_mechanism.csv`, `e1_gated.csv`
Coverage and widths	36 settings × 20 designs × 100 draws per d	`v14_recommended.csv`, `v12ab_all_cells.csv`
Coverage on reported values (post hoc)	same draws as above	`posthoc/v14_guard_` `coverage.csv`
Merging on HotpotQA	pooled HotpotQA, 100 questions × 24 passes, $n = 992$, $T = 2{,}400$, `aopt0.4`	`x1_merge.csv`
Merging on LoCoMo	two conversations, 304 questions, $n = 788$, $T = 912$, three passes, score perturbation $\beta = 0.08$, 300 sampled columns	`locomo_` `identifiability.csv`

E Benchmark Audit

This appendix gives the protocol of the audit in Section 5.2 (Appendix E.1), proves that every one of its 55 configurations determines no memory value (Appendix E.2), and describes each benchmark (Appendix E.3).

E.1 Protocol

For each benchmark, we build its store as the benchmark specifies and run its retrieval once over its questions, one episode per question, at each retrieval size it uses. We add the sizes 3, 5 and 10, at the size it specifies where that differs, and at all for the four long-context benchmarks. Each configuration gives a log $S_1, \ldots, S_T$, from which we build $\mathbf{X}_1$ and test every memory with Algorithm 1 at $d = 1$, with the baseline column included. We also compute the blocks of the log (Definition A.5), the mean retrieval size, and the number of memories never retrieved. The audit uses the retrieved sets and no outcome, so it calls no reader: by Appendix A.1, whether a value is determined does not depend on the outcomes. By Lemma B.3, a memory undetermined at $d = 1$ is undetermined at every d, so the audit covers every resolution.

E.2 WHY EVERY CONFIGURATION DETERMINES NOTHING

The audit finds no determined value in any of the 55 configurations (Table 2). The following lemma explains all of them at once. It extends Theorem 1 from one retrieval size to one size per block.

Lemma E.1 (Constant size within each block)**.** *Suppose that, within each block b of a log, every episode retrieves the same number k_b of memories. Then no memory is identifiable at any resolution.*

Proof. Define $\mathbf{w} \in \mathbb{R}^n$ by $w_i = 1/k_b$ for every memory i of block b, and $w_i = 0$ for memories never retrieved. An episode of block b retrieves k_b memories, all in block b, so $(\mathbf{X}\mathbf{w})_t = k_b \cdot (1/k_b) = 1$ for every t. Hence $(-1, \mathbf{w}) \in \ker \mathbf{X}_1$, and for every retrieved memory i the inner product of $(0, \mathbf{e}_i)$ with this vector is $w_i \neq 0$. A memory never retrieved has a zero column, so $(0, \mathbf{e}_i) \in \ker \mathbf{X}_1$ and its inner product with itself is 1. No memory is identifiable at $d = 1$, and by Lemma B.3 at no higher resolution. □

We verified its hypothesis on every one of the 55 configurations. The hypothesis holds in every configuration of the audit. At a fixed k, every episode of the log retrieves k memories. At "all", every episode of a context retrieves the whole context, so the size is constant within each context, and each block lies within one context. The vector $(-1, \mathbf{w})$ is a change like $\mathbf{h}_\emptyset$: it lowers the baseline and raises each memory's value by the inverse of its block's retrieval size.

Blocks limit the log further, and would limit it even if the sizes varied. In the six benchmarks with a store per question and one episode per question, every block has $T_b = 1$, so by Proposition 2 at most one value per question could be determined, whatever the retrieval. The larger HotpotQA run, with 1,200 questions, confirms that more questions do not help: the number of blocks grows with the number of questions, and every block still has one episode. The two benchmarks with a shared store, MultiHopRAG and HELMET, split into fewer and larger blocks: MultiHopRAG from 269 blocks at $k = 3$ to 15 at $k = 10$, and HELMET into 41 blocks of 8.7 episodes on average at every k. Their blocks have several episodes each, and the fixed size alone leaves every value undetermined. A shared store is therefore necessary and not sufficient, as Section 5.2 states.

E.3 THE BENCHMARKS

Table 5 gives, for each benchmark, the unit we use as a memory, the scope of its store and the retrieval sizes of the audit. The size the benchmark itself specifies is in bold. The counts T, n and the blocks are in Table 2.

Three details affect the counts. In HotpotQA, $n = 2{,}971$ for 300 questions because a few validation items ship fewer than ten paragraphs; at $k = 10$, every paragraph of a question is retrieved. In the "all" configurations, the log keeps T equal to the number of questions and changes only the retrieval size, whose mean is 104.0 for BABILong, 942.4 for InfiniteBench, 125.5 for QuALITY and 1,372.5 for NarrativeQA on the full text. The HELMET store is capped at 40,000 passages by us, for compute.

F ADDITIONAL RESULTS

This appendix reports the full results behind Section 5, in its order: retrieval on pooled HotpotQA (Appendix F.1), the replications on LoCoMo and LongMemEval (Appendix F.2), the mechanism of the accuracy cost (Appendix F.3), additional readers (Appendix F.4), intervals (Appendix F.5), merging (Appendix F.6), and the uses of the determined values (Appendix F.7). Appendix D specifies every run, and Table 4 names its artifact.

F.1 RETRIEVAL ON POOLED HOTPOTQA

Table 6 gives every point of Figure 3a. The fraction determined is computed on 300 questions over 24 passes ($n = 2{,}971$, $T = 7{,}200$) and averaged over five seeds; LEDGER's accuracy is measured on 1,600 held-out questions, and the random rules' on 300 questions with Holm-corrected tests.

Table 5: The fourteen audited benchmarks. "All" retrieves every memory of the context; bold marks the size the benchmark specifies, where it specifies one.

Benchmark	Memory	Store	Sizes
HotpotQA (Yang et al., 2018)	paragraph	per question	3, 5, 10
MuSiQue (Trivedi et al., 2022)	paragraph	per question	3, 5, 10
WikiHop (Welbl et al., 2018)	supporting document	per question	3, 5, 10
TriviaQA (Joshi et al., 2017)	paragraph of the Wikipedia contex	per question	3, 5, 10
LongMemEval (Wu et al., 2025)	turn	per question	3, 5, 10
MSC-Self-Instruct (Xu et al., 2021)	turn	per dialogue	3, 5, 10
BABILong (Kuratov et al., 2024)	sentence	per item	3, 5, 10
InfiniteBench (Zhang et al., 2024)	paragraph, first 1,000 per book	per book	3, 5, 10
NarrativeQA, summary (Kočiskỳ et al., 2018)	paragraph of the full text	per document	3, 5, 10
NarrativeQA, full text	chunk of the full text	per document	3, 5, 10
QuALITY (Pang et al., 2022)	paragraph	per article	3, 5, 10
LoCoMo (Maharana et al., 2024)	turn	per conversation	3, 5, 10
MemoryAgentBench (Hu et al., 2026)	paragraph	per context	3, 5, 10
MultiHopRAG (Tang & Yang, 2024)	paragraph	shared	3, 5, **6**, 10
HELMET (Yen et al., 2025)	passage	shared, 40,000	3, 5, 10, **14**

Table 6: Retrieval rules on pooled HotpotQA. Fractions determined are with the baseline column fitted and, in parentheses, without it; accuracy changes are against deterministic top-k, in points, with 95% intervals in brackets. LEDGER's accuracy is measured on 1,600 questions; the random rules' on 300, with Holm-corrected p-values.

Rule	Parameter	Determined	Mean $\lvert S_t \rvert$	Accuracy change	p
Top-k	—	0.001 (0.003)	5.33	—	—
LEDGER	$\rho = 0.4$, $M = 20$	0.797 (0.797)	5.25	-2.6 $[-4.2, -0.9]$	0.004
ε-greedy	$\varepsilon = 0.1$	0.001 (0.003)	5.33	-3.0 $[-5.3, -1.0]$	0.012
ε-greedy	$\varepsilon = 0.3$	0.989 (0.989)	5.33	-13.3 $[-17.3, -9.7]$	< 0.001
Perturbation	$\beta = 0.05$	0.983 (0.983)	5.33	-14.0 $[-19.3, -8.7]$	< 0.001
Perturbation	$\beta = 0.10$	1.000 (1.000)	5.33	-37.0 $[-43.0, -30.7]$	< 0.001

The fraction determined under ε-greedy with $\varepsilon = 0.1$ equals that of top-k with and without the baseline column. The baseline-test residual is 3.1×10^{-15} under deterministic top-k, between 2.8×10^{-15} and 3.4×10^{-15} across the five seeds, and 9.2×10^{-2} under LEDGER's rule.

F.2 Replications on LoCoMo and LongMemEval

Table 7 gives the accuracy comparison on the two other corpora, with the same rules and sizes. On LoCoMo, LEDGER raises accuracy; on LongMemEval, the change is not distinguishable from zero.

Table 7: Accuracy of deterministic top-k and LEDGER on LoCoMo and LongMemEval, with the change in points and its 95% interval.

Corpus	Questions	Top-k	LEDGER	Change	95% interval
LoCoMo	1,542	0.191	0.217	+2.7	[+1.0, +4.4]
LongMemEval	479	0.263	0.271	+0.8	[−1.9, +3.8]

On LoCoMo we also compare the fraction determined across rules, on two conversations with $n = 788$ memories and 120 questions over 12 passes ($T = 1{,}440$). Deterministic top-k determines 0.193 of the memories, ε-greedy with $\varepsilon = 0.3$ determines 0.217, LEDGER 0.321, and score perturbation 0.445 at $\beta = 0.05$ and 0.532 at $\beta = 0.10$. On this store, deterministic top-k determines more than on pooled HotpotQA, and LEDGER determines less than score perturbation.

F.3 The mechanism of the accuracy cost

Table 8 gives the split of Figure 3. On pooled HotpotQA, top-k retrieves 0.93 of the gold memories, and LEDGER lowers the number of gold memories retrieved per question from 2.000 to 1.758 on questions where top-k retrieved both.

Table 8: LEDGER against top-k on pooled HotpotQA, split by whether top-k retrieved every gold memory of the question.

Questions	Share	Gold lost per question	Accuracy change	95% interval
Top-k found all gold	0.42	0.242	−6.9 [−9.6, −4.3]	0.242
Top-k missed some gold	0.58	0.009	+0.7 [−1.4, +2.7]	0.009

The pre-registered query-time substitute applies LEDGER when the query's top-1 similarity is below its median over the 1,600 questions, 0.647. It changes accuracy by −1.3 points ([−2.6, −0.1]) against top-k and by +1.25 ([0.0, +2.4]) against ungated LEDGER, and determines 0.520 of the memories. The pre-registered rule required the second interval to exclude zero, so the gate does not transfer the oracle's result (Appendix G).

F.4 Pooling across readers

Table 9 repeats the comparison of LEDGER and top-k on pooled HotpotQA with two further readers. The retrieval logs, and therefore every fraction determined, are the same for all readers; only the accuracies change.

Table 9: Accuracy of three readers on LongMemEval when each question retrieves from its own store and from one pooled store, with the cost of pooling in points and its 95% interval; 300 questions per cell. The gold column gives the reader only the evidence sessions.

Reader	Gold	Own store	Pooled store	Cost of pooling
`gemini-2.5-flash-lite`	0.607	0.417	0.260	15.7 [11.0, 20.7]
`gemini-3.6-flash`	0.783	0.497	0.287	21.0 [15.7, 26.3]
`gemini-3.7-flash`	0.757	0.477	0.280	19.7 [14.7, 25.0]

F.5 Intervals

Table 10 gives joint coverage at $\delta = 0.05$ on the synthetic logs of Appendix D.4, as the minimum over settings. Coverage over every memory counts the intervals of undetermined memories, which LEDGER does not report; coverage over reported values counts only the intervals that Algorithm 1 returns.

Table 10: Lowest joint coverage over settings at $\delta = 0.05$, with the target 0.95. The last column restricts coverage to the values LEDGER reports; it is a post hoc analysis.

d	Designs	Settings	Settings reporting	Every memory	Reported values
1	full rank	36	36	0.977	0.977
2	full rank	12	12	0.994	0.994
2	rank-deficient	24	6	0.572	0.995

In the 18 rank-deficient settings that report nothing, every memory of every design is undetermined. Their coverage over every memory ranges down to 0.572, and adding the exact ridge-bias term of Proposition 3, which needs the true $\boldsymbol{\theta}$, raises the minimum over all 36 settings at $d = 2$ to 0.992. This confirms the explanation of Appendix B.7: the undercoverage comes from the part $\mathbf{c}_2^\top \boldsymbol{\theta}$ of undetermined values. The coverage of a design does not predict whether it reports: one rank-deficient setting ($n = 30$, $k = 3$, $T = 300$) has coverage 0.976 over every memory and reports nothing, because every memory there has a component in the kernel. A separate grid at $\lambda = 1$ contains one rank-deficient design at $d = 1$, with coverage 0.964 over $20{,}000$ draws.

Table 11 compares widths. LEDGER's interval is close to the pointwise Gaussian interval that knows σ; the self-normalised confidence set is several times wider at $d = 1$ and an order of magnitude wider at $d = 2$, because its width grows with the number of dividends.

Table 11: Width of each interval relative to the pointwise Gaussian interval with the true σ: median over settings, with the range.

Interval	$d = 1$	$d = 2$
LEDGER, Eq. Eq. (5)	1.52	1.43
Abbasi-Yadkori et al. (2011), $\lambda = 1$	6.6 (4.7–9.4)	35.9 (11.0–365.7)

F.6 Merging

Table 12 gives the merging results on pooled HotpotQA with 100 questions over 24 passes ($n = 992$, $T = 2{,}400$), for two retrieval rules. Fractions are over memories before merging and over units after it.

Table 12: Units and fraction determined before and after merging on pooled HotpotQA.

	Log		Identical histories		Similar histories	
Rule	Units	Determined	Units	Determined	Units	Determined
LEDGER	992	0.815	869	0.992	867	0.995
Perturbation, $\beta = 0.05$	992	0.969	964	0.999	964	0.999

Merging identical histories removes 123 units under LEDGER and 28 under score perturbation, which ties fewer memories. Merging similar histories, with candidates that differ in at most eight episodes, accepts two merges under LEDGER and none under perturbation. On LoCoMo, after three passes of score perturbation with $\beta = 0.08$, 79.0% of the memories are determined, and merging raises this to 95.7% of the units, both estimated on 300 randomly sampled columns.

F.7 Uses of the determined values

Value recovery. On synthetic logs with 40 memories and $T = 4{,}000$, over 30 seeds, the Spearman correlation with the true Shapley values is 0.892 for LEDGER on the 31 memories it reports, 0.804 when LEDGER splits each group total equally to score all 40, 0.724 for reward averaging and 0.118 for retrieval frequency.

Retention under a budget. Ranking undetermined memories last and keeping 30% of the store, the retained store succeeds in 5.52 of 300 evaluation episodes under LEDGER, 6.55 under the equal split, 7.14 under reward averaging and 6.48 under the true Shapley values; the pre-registered real-data retention test was not completed.

G Pre-registration and Corrections

This appendix records how the empirical claims were tested: which were pre-registered and with what decision rule (Appendix G.1), which were withdrawn (Appendix G.2), the one analysis that was not pre-registered (Appendix G.3), an amendment on the name of the retrieval rule (Appendix G.4), and the defects we found in our own pipeline with the values they changed (Appendix G.5). The dated documents are in the `preregistration/` directory of the released code, and the errata in `docs/errata.md`.

G.1 Pre-registered claims

Before each measurement, we recorded the claim, the quantity that tests it and the rule that decides it, in a dated document committed to the repository. Table 13 lists the claims and their outcomes. A claim holds when its measured quantity, with its interval, satisfies the rule; otherwise it is withdrawn and reported as such.

G.2 Withdrawn claims

Three pre-registered claims were withdrawn because their measured values did not meet their decision rules. We report each with the value it was measured at.

1. *Instrumentation is approximately free.* The rule required the interval on LEDGER's accuracy change against top-k to contain zero with a half-width below 0.025. On $1{,}600$ questions of pooled HotpotQA the change is -2.6 points ($[-4.2, -0.9]$, permutation $p = 0.004$), so the paper reports a cost (Section 5.3).
2. *The cost of pooling grows with reader quality.* The rule required the pairwise differences between readers to exclude zero. The costs are 15.7, 21.0 and 19.7 points, and two of the three pairwise intervals contain zero (Appendix F.4).
3. *The sample-complexity bound.* Validation check V7 required the fitted slope of the error against the bound to be at least 1.00; it is 0.82, which is why 21 of the 22 checks pass.

Four further pre-registered tests had negative or incomplete outcomes and are reported in Table 13: the remedy on three audited benchmarks, the reach ceiling, the label-free gate and the retention comparison. The gate was pre-registered as a query-time substitute for the oracle of Section 5.3. It applies LEDGER when the query's top-1 similarity is below its median, and it did not transfer the oracle's result (Appendix F.3).

G.3 The post hoc analysis

One result in the paper was not pre-registered: joint coverage restricted to the values that LEDGER reports (Section 5.4 and Table 10). The pre-registered coverage test counted the intervals of every memory, including undetermined ones that LEDGER does not report, and at $d = 2$ it fell to 0.572 on rank-deficient designs. After seeing this, we restricted coverage to the reported values, which gives a minimum of 0.994 over all 36 settings at $d = 2$. The restriction was chosen after the failure was observed, so it is an analysis and not a test. It uses the same seeds, settings and draws as the pre-registered test, whose numbers are unchanged. Its script and output are released separately from the pre-registeredchecks, and the errata record it as post hoc with its date.

G.4 Amendment on the retrieval rule

One pre-registration document describes LEDGER's retrieval rule as A-optimal tie-breaking. The rule that was run is Eq. (8): it fills $\max(1, \mathrm{round}(\rho k_t))$ places with the least-retrieved of the M most

Table 13: Pre-registered claims, their decision rules and outcomes, in the order of the pre-registration documents. "Where" gives the section that reports the result; a dash marks a claim reported only in the released documents.

Claim	Decision rule	Where	Outcome
Standard benchmarks determine no memory value	falsified if a private-store protocol gives fewer than 0.9 times as many blocks as contexts, or a fraction above 0.01 at a fixed k	5.2	held: 0 of 55 configurations
The remedy recovers identifiability on three audited benchmarks	fractions above floors of 0.50, 0.22, 0.13; falsified below 0.10 or a quarter of the rank bound	F	failed: 0.46, 0.13, 0.08; falsified on MemoryAgentBench
Reach is the binding ceiling	falsified if the window arm is flat or non-monotone, or the passes arm rises by 0.05 or more	—	failed on two of three benchmarks
The crossover rule on synthetic logs	rises below $R = T$, falls above, peak in $[0.5, 2]$, range at least 0.02	—	held: peak at $R/T = 0.75$, range 0.14
Instrumentation is approximately free	interval contains zero with half-width below 0.025	5.3	withdrawn: -2.6 $[-4.2, -0.9]$
Displacement is the mechanism	gold retrieved per question on found-gold questions below 2.00	5.3	held: 1.758
The effect's sign differs by partition	negative where top-k found all gold; non-negative elsewhere	5.3	held: -6.9 $[-9.6, -4.3]$; $+0.7$ $[-1.4, +2.7]$
A label-free gate transfers the oracle's result	interval of gated minus ungated excludes zero	F.3	failed: $+1.25$ $[0.0, +2.4]$
The pooling cost grows with store size	interval of largest minus smallest store excludes zero	—	held at 100 questions ($+11$ $[+4, +18]$); null at 300 ($+2$ $[-2, +6]$)
The pooling cost grows with reader quality	pairwise differences between readers exclude zero	F.4	withdrawn: two of three contain zero
Retention by LEDGER loses less than retention by frequency	paired interval at a 50% budget excludes zero	—	not completed
The sample-complexity bound (check V7)	fitted slope at least 1.00	—	withdrawn: slope 0.82
The validation suite	each check's pass criterion	—	21 of 22 checks pass; V7 fails

similar memories and the rest by similarity. It is motivated by A-optimal design (Appendix A.5) and optimises no criterion, and we did not compare it with an A-optimal choice. The pre-registration is left verbatim, and a dated amendment records this; the name is the only thing at issue, and no number changes. The code and its artifacts keep the label `aopt0.4` for this rule.

G.5 Errata

Table 14 lists the defects we found in our pipeline and in the reporting of its results, with the values they affected. Every value in the main text is the corrected one.

H Discussion

H.1 Interpreting the results

What the results add to estimability. The condition that decides identifiability is classical: a linear functional $\mathbf{c}^\top \boldsymbol{\theta}$ is estimable exactly when $\mathbf{c}$ lies in the row space of the design matrix (Rao, 1973). The paper adds three things. First, for a retrieval log the design is determined by the retrieved sets, and its Gram matrix at $d = 1$ consists of the retrieval counts Eq. (3), so the condition can

Table 14: Defects found and the values they changed. Every value in the paper is the corrected one; each entry in `docs/errata.md` gives the full account.

Defect	Earlier value	Corrected value
Pipeline		
Baseline offset mishandled in eight estimator methods and four callers	coverage 0.102; value recovery at chance	coverage 1.000; recovery restored
Three bugs in the merging rule for undetermined totals (Case 1)	60 of 60 accepted	17 and 25 of 60
A shadowed tolerance and a ridge-inverted kernel in one validation test, and a fixed-size test that divided by zero	tests passed without measuring	tests measure coverage
Paragraph splitter fell back silently	NarrativeQA as 11 memories; InfiniteBench as 4 episodes	115 and 14,282 memories; 40 episodes
Audit rows with $\operatorname{rank} \mathbf{X} = T$ read as evidence	29 of 55 counted	reported as automatic
Cross-item figure derived from retrieval	94.1%	withdrawn; replaced by block counts
Store-size sweep built on prefixes of one store	cost appeared to shrink	cost rises with store size
Synthetic configuration changed on 2026-08-16	retention table unreproducible	regenerated from the current configuration
Two artifacts overwritten; one path-resolution incident	—	regenerated and guarded
Sample size and attribution		
LEDGER's accuracy cost at 300 questions	interval containing zero	-2.6 $[-4.2, -0.9]$ at 1,600
LoCoMo accuracy change at 120 questions	$+6.7$	$+2.7$ at 1,542
Pooling cost not attributed to a reader	16.7	15.7, 21.0, 19.7 by reader
Reporting, corrected before submission		
Width of the self-normalised set taken from single settings	$6.5\times$ and $96\times$	$6.6\times$ and $36\times$ (median)
LoCoMo merging run attributed to LEDGER's retrieval	LEDGER, three conversations	score perturbation, two conversations
Coverage settings described as designs	36 designs	36 settings of 20 designs
Extra readers described as repeating the LEDGER comparison	LEDGER against top-k	cost of pooling only
Coverage described with the estimated noise scale	$\hat{\sigma}$	known $\sigma = 1$

be checked from counts that every agent already records. Second, the structure of these counts under common conventions forces specific invisible changes. A fixed retrieval size places $\mathbf{h}_B$ in $\ker \mathbf{X}_d$ for every $|B| < d$ (Theorem 1). Since $(\mathbf{c}^{(i)})^\top \mathbf{h}_\emptyset = 1$ for every memory at $d = 1$, no memory is identifiable at any number of episodes. For binary outcomes the conclusion is stronger than the absence of an unbiased linear estimator: the distribution of the log is the same for two different values of φ_i, so no estimator is consistent. Third, the condition fails on every one of the 55 configurations of fourteen standard benchmarks (Table 2). Methods that assign a value to every memory therefore report, on these logs, numbers that the solver chooses and the data do not constrain. Slate off-policy evaluation works with the same counts and sets the undetermined part aside, because it does not change the value of a policy (Swaminathan et al., 2017); for memory valuation, the undetermined part names the memories that cannot be valued.

Why Shapley values and Möbius dividends. The Shapley value is the unique attribution that satisfies efficiency, symmetry, null player and additivity (Shapley, 1953), and it is the notion used by attribution-based memory systems (Wei et al., 2026b; Ye & Yoganarasimhan, 2025; Patel et al., 2025). Writing v in dividends makes the expected outcome of each episode linear in $\boldsymbol{\theta}$ and makes each value a linear functional $(\mathbf{c}^{(i)})^\top \boldsymbol{\theta}$, which is what turns valuation into a question about the row

space of $\mathbf{X}_d$. The same test applies to any value that is linear in $\boldsymbol{\theta}$, with its own coefficient vector. The Banzhaf value (Banzhaf III, 1964), for example, is $\sum_{B \ni i} 2^{1-|B|}\theta_B$. At $d = 1$ the choice does not matter: every marginal contribution $v(S \cup \{i\}) - v(S)$ equals $\theta_{\{i\}}$, so every probabilistic value (Weber, 1988) of memory i equals $\theta_{\{i\}}$, and our results at $d = 1$ hold for all of them.

Resolution. The experiments on real logs test identifiability at $d = 1$, and this suffices for the audit. If $\mathbf{h} \in \ker \mathbf{X}_d$ and $(\mathbf{c}^{(i)})^\top \mathbf{h} \neq 0$, extending $\mathbf{h}$ by zeros on the columns of order $d + 1$ gives a vector in $\ker \mathbf{X}_{d+1}$, and its inner product with the coefficients of φ_i at resolution $d + 1$ is the same nonzero number, because the two coefficient vectors agree on sets of size at most d. A memory undetermined at resolution d is therefore undetermined at every higher resolution, and the 55 configurations determine no value at any d. The positive results use $d = 1$ for a different reason. The design $\mathbf{X}_d$ has $p_d = \sum_{\ell=0}^{d} \binom{n}{\ell}$ columns and rank at most T, and the log needs at least $d + 1$ distinct retrieval sizes (Proposition 1); at $d = 2$ and $n = 2{,}971$, $p_2 \approx 4.4 \times 10^6$, far above $T = 7{,}200$. We therefore study $d = 2$ on synthetic logs only (Section 5.4). What $d = 1$ omits is bounded by Proposition A.1: $|\varphi_i(v) - \varphi_i(v^{(1)})| \leq \frac{1}{2} \sum_{B \ni i,\, |B|>1} |\theta_B|$.

Associational and causal values. Identifiability depends on $\mathbf{X}_d$ alone, and so do all results of Section 3. Estimation depends also on the noise. Writing $v_q = v + \delta_q$, the noise of the linear model is $\varepsilon_t = \eta_t + \delta_{q_t}(S_t)$, and because retrieval reads the query, $\delta_{q_t}(S_t)$ can correlate with the design. The estimates then describe the log: they equal the dividends of v when the log is exogenous (Definition A.3), for example when S_t is independent of q_t, and they can be biased otherwise. Appending query features to the design removes the part of $\boldsymbol{\delta}$ in their span and leaves every result on identifiability unchanged, since appending columns never makes a non-estimable functional estimable (Appendix A.3). A separate condition concerns adaptivity: Proposition 3 requires that the retrieved sets do not depend on past outcomes. LEDGER's rule Eq. (8) depends on the past only through the counts $f_i(t)$, which are functions of $\mathbf{X}$, so it satisfies this condition.

Random retrieval. Randomising retrieval also breaks the nesting of top-k, and at high enough rates ε-greedy and score perturbation determine more values than LEDGER (Figure 3a). They draw replacements from the whole store, so the reader loses evidence: reaching a fraction of 0.98 or more costs them between 13.3 and 37.0 accuracy points. LEDGER draws its replacements from the M memories most similar to the query and keeps the most similar ones in H_t, which bounds how much evidence it can displace.

The accuracy cost. LEDGER's effect on accuracy ranges from -2.6 points on HotpotQA to $+2.7$ on LoCoMo and falls as the gold recall of top-k rises (Figure 3b). Its mechanism is displacement: the loss comes from questions where top-k had already retrieved every gold memory (Figure 3c). Where top-k misses evidence, spreading retrieval costs nothing measurable, and an oracle that applies LEDGER only there keeps 64% of the gain with no measurable loss. The pre-registered query-time substitute, top-1 similarity, did not reproduce this result (Appendix F).

H.2 Scope and open problems

Scope. The analysis uses three properties of the setting: in each episode a system chooses a set of items from a pool, it observes one scalar outcome, and the expected outcome is a sum of dividends of order at most d. It does not use the fact that the items are memories. The same test therefore applies to the tools an agent calls in an episode, the passages a retrieval-augmented generator reads, and the slates a recommender shows (Swaminathan et al., 2017), and each of these settings has its own versions of the fixed size, the private pool and the tied items.

A query-time gate. The oracle of Section 5.3 shows that most of the gain is available without cost when LEDGER is applied only to questions where top-k misses evidence. A deployed agent has no gold labels, and the first substitute we pre-registered failed. A gate that predicts, at query time, whether top-k has found the evidence would make this trade available in deployment.

An exact optimal-design rule. LEDGER's rule Eq. (8) targets the diagonal of $\mathbf{V}^{-1}$ through retrieval counts, using $(\mathbf{V}^{-1})_{ii} \geq 1/V_{ii}$. The A-optimal choice (Pukelsheim, 2006) among sets drawn

from R_t would maximise the reduction in $\mathrm{tr}(\mathbf{V}^{-1})$, which by the Sherman–Morrison identity is $\|\mathbf{V}^{-1}\mathbf{x}_S\|^2/(1+\mathbf{x}_S^\top\mathbf{V}^{-1}\mathbf{x}_S)$ for a set with row $\mathbf{x}_S$. A greedy search over R_t costs $O(Mk_tp_d)$ per episode given $\mathbf{V}^{-1}$, and a rank-one update of $\mathbf{V}^{-1}$ costs $O(p_d^2)$. Whether it determines more values than the count-based rule, and at what cost in accuracy, is open.

Higher resolution. At $d \geq 2$, the number of dividends grows as n^d, and most values need T of the same order to be determined. If only a few dividends of order two are nonzero, identifiability becomes a question of sparse recovery, whose conditions on $\mathbf{X}_d$ differ from the row-space condition used here.

Other outcome models. For a binary outcome with an invertible link g, $\mathbb{E}[r_t] = g^{-1}\big((\mathbf{X}_d\boldsymbol{\theta})_t\big)$, the distribution of the log determines $\mathbf{X}_d\boldsymbol{\theta}$, so the row-space condition and every result of Section 3 carry over unchanged. The dividends then describe the value on the scale of the link, and the estimator and its intervals would need to change.

Adaptive and changing stores. Rules that choose sets from past outcomes, such as bandit algorithms, violate the condition of Proposition 3. Intervals that remain valid under such rules exist, such as the confidence set of Abbasi-Yadkori et al. (2011), but they are 6.6 to 36 times as wide in our synthetic study. A store that grows over time adds columns that start at zero: a new memory is undetermined until it has been retrieved with varied partners and at varied sizes, which gives a natural schedule for when its value can first be reported.

H.3 Applications

Attribution in language models. Attribution asks which parts of a model's input or training data an output depends on. Data attribution values training examples by their effect on a trained model (Ghorbani & Zou, 2019), context attribution values the sources in a prompt by their effect on a generation (Cohen-Wang et al., 2024), and Shapley-based methods value the passages of a retrieval-augmented generator (Patel et al., 2025; Ye & Yoganarasimhan, 2025). These methods obtain their evidence by design: they ablate random subsets of the sources and fit a surrogate to the outcomes. A deployed system cannot run such ablations on its own traffic, and when the same estimators are fitted to logged data, they return a number for every source whether or not the log determines it. Our results supply the missing check. Whether a log determines an attribution depends only on the log's counts, so it can be decided before any attribution is estimated, and it applies to every attribution that is linear in the dividends (Appendix H.1). The results also explain why designed ablations succeed: independent random subsets vary in size and are not nested, so with enough of them $\mathbf{X}_1$ has full column rank, while the logs that retrieval produces under a fixed size have a nonzero kernel at every T (Theorem 1).

Cognitive models of memory. The rational analysis of human memory holds that the availability of a memory tracks the probability that it will be needed, which the environment reveals through the frequency and recency of past use (Anderson & Schooler, 1991). Agent memory systems adopted these signals directly: they score memories by recency, importance and relevance (Park et al., 2023), decay them along a forgetting curve (Zhong et al., 2024), or rank them by access frequency (Wei et al., 2026a; Wu et al., 2026a). Frequency and recency measure how often a memory is needed. They do not measure how much a memory contributes to the outcome when it is retrieved, and a store managed for an agent needs both. Our results separate the two quantities within one object. The retrieval frequencies are the diagonal of $\mathbf{X}^\top\mathbf{X}$ Eq. (1), so they are exactly what a log records about need, while the contribution φ_i is a functional of the outcomes. The same counts decide whether the contribution can be learned: a memory retrieved in every episode of a top-k log has the largest frequency and an undetermined value (Theorem 1). A model of agent memory built on need alone therefore cannot tell a frequently retrieved memory that helps from one that is only retrieved often, and LEDGER supplies the missing quantity where the log determines it.

Human control of personal memory. Assistants with persistent memory store facts about their users across conversations. Users value the personalization this brings, and they ask for transparent control over what is stored and for accurate memories (Chen et al., 2026a). Control requires knowing which memories matter. Research on human–AI decision-making shows that people should rely on

a system's advice only when it is correct (Schemmer et al., 2023), and that over-reliance on incorrect advice resists simple remedies such as explanations (Buçinca et al., 2021). A user deciding whether to delete, correct or keep a memory relies on the system's account of that memory, and current systems show what is stored without any statement of what each memory does. LEDGER gives each memory one of two answers that a user can rely on. A determined value comes with a confidence interval, so its uncertainty is stated. An undetermined value is reported as such, so the system makes no claim that its log cannot support. Merging reduces the store without losing a determined value (Theorem 2), which makes it safe to present fewer, larger units to a user. The same statements identify which memories an agent relies on, which supports profiling as well as control, and they should be governed by the policy that governs the store.

Training memory agents with reinforcement learning. Reinforcement learning from outcome rewards underlies recent reasoning models (Shao et al., 2024; Guo et al., 2025), and it now trains agents to manage memory: which entries to add, update or delete, and which to use in an answer (Yan et al., 2026b). Related methods learn a value for each memory from downstream reward (Tan et al., 2025; Zhang et al., 2026; Yang et al., 2026). Every such method faces a credit-assignment problem: the reward belongs to the episode, and the value of each memory must be inferred from which memories the episode retrieved. This is the inverse problem of this paper, with the rollouts as the log and the reward as the outcome. When rollouts retrieve a fixed number of memories, the rewards leave every memory's credit undetermined (Theorem 1), so a learned per-memory value is fixed by the learner and not by the rewards; co-retrieved memories then receive spurious credit (Yang et al., 2026; Yan et al., 2026a). Our test applies to the rollout log before training, because it reads only which memories each rollout retrieved. LEDGER's retrieval makes rollouts that determine more values, and its determined values, with their intervals, can serve as per-memory rewards that no undetermined memory receives. Rollout policies that adapt to past rewards violate the condition of Proposition 3, so their intervals need the wider self-normalized bounds of Abbasi-Yadkori et al. (2011); identifiability is unaffected, since it depends on $\mathbf{X}_d$ alone.

Evaluating memory systems. Memory benchmarks have moved from single questions to long conversations and multi-session interactions, and from answer accuracy toward the operations a memory system performs over time (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2026). They still score a memory system by the answers it produces. None of them can score its valuation, because none of them provides ground truth for the value of a memory, and their own protocols leave every value undetermined (Table 2). Our results provide ground truth of two kinds. Whether a value is determined is a theorem about the log, so a benchmark can report the fraction of values its logs determine alongside accuracy, and a valuation method can be tested on whether it abstains exactly on the undetermined memories. Where a value is determined, synthetic logs with known dividends test its interval, as in Section 5.4. The changes a benchmark needs are small and are all changes to its protocol: a shared store, several retrieval sizes and several passes per question.

H.4 Limitations

Model. The results assume that the query-averaged value has no dividends above order d (Assumption 2), and the real-data experiments use $d = 1$. Omitted dividends bias the estimates wherever they correlate with the design, and Proposition A.1 bounds only their direct effect on each value. The noise is assumed conditionally sub-Gaussian (Assumption 1). The reported values are associational unless the log is exogenous (Appendix H.1).

Estimation. Coverage of the intervals is measured on synthetic logs drawn from the model, where the true values are known; on real logs it cannot be measured. The result that coverage holds on every reported value at $d = 2$ comes from a post hoc analysis (Appendix G). The interval in Eq. (5) omits the ridge-bias term of Proposition 3, which is small for $\lambda = 10^{-2}$ on the values LEDGER reports and is the cause of undercoverage on values it does not report.

Retrieval. LEDGER's rule is a count-based heuristic motivated by A-optimal design; it optimises no criterion. Its sets fall below the drawn size in 6.7% of episodes on pooled HotpotQA (Appendix D). The rule treats the $M = 20$ memories most similar to the query as candidates of comparable relevance, and we did not test this assumption directly. Its effect on accuracy depends on

the corpus, from -2.6 to $+2.7$ points, and the only rule that removes the cost is an oracle that needs gold labels.

Experiments. The accuracy experiments use hosted readers that cannot be pinned and a single judge model, so they are reproducible in design and not bit for bit. The pooled stores for HotpotQA and LongMemEval are our construction; the benchmarks define private stores. The audit runs each benchmark at one pass. The LoCoMo merging result uses score perturbation, not LEDGER's rule, on three conversations, with fractions estimated on 300 sampled columns. The random rules' accuracy is measured on 300 questions, and LEDGER's on $1{,}600$.

I Notation and Terms

Vectors are bold lowercase and matrices bold uppercase. We write $[n] = \{1, \dots, n\}$, $\mathbf{1}_T$ for the all-ones vector in $\mathbb{R}^T$, and $\mathbf{I}$ for the identity. For a matrix $\mathbf{A}$, $\operatorname{range} \mathbf{A}$ is its column space, $\operatorname{range} \mathbf{A}^\top$ its row space and $\ker \mathbf{A}$ its kernel. Memory m_i is identified with its index i, so $S_t \subseteq [n]$. Table 15 lists the symbols, and Table 16 defines the terms.

Table 15: Symbols, in the order they appear.

Store, episodes and retrieval	
n	number of memories in the store
m_i	memory i, for $i \in [n]$
T	number of episodes
$q_t,\ \mathcal{D}_Q$	query of episode t; the distribution it is drawn from
S_t	set of memories retrieved in episode t
$k_t = \lvert S_t \rvert$	retrieval size of episode t; written k when every episode retrieves the same number
$r_t,\ \mathbf{r}$	outcome of episode t; the vector of outcomes
$\mathcal{F}_{t-1}$	history before episode t
Model	
v_q	expected outcome for query q as a function of the retrieved set (Assumption 1)
v	query-averaged value, $v(S) = \mathbb{E}_q[v_q(S)]$
η_t	$r_t - v_{q_t}(S_t)$, conditionally σ-sub-Gaussian
ε_t	$r_t - v(S_t)$, the noise of the linear model Eq. (2)
σ	sub-Gaussian noise scale
$\theta_B,\ \boldsymbol{\theta}$	Möbius dividend of the set B; the vector of dividends with $\lvert B \rvert \le d$
$\theta_\emptyset$	baseline: the expected outcome with nothing retrieved
d	resolution: $\theta_B = 0$ for $\lvert B \rvert > d$ (Assumption 2)
φ_i	value of memory i: its Shapley value under v, $\sum_{B \ni i} \theta_B / \lvert B \rvert$
$\mathbf{c}^{(i)}$	coefficients with $\varphi_i = (\mathbf{c}^{(i)})^\top \boldsymbol{\theta}$: $c_B^{(i)} = \mathbf{1}[i \in B]/\lvert B \rvert$
Log and design	
$\mathbf{X}$	$T \times n$ binary matrix, $\mathbf{X}_{ti} = \mathbf{1}[i \in S_t]$
$\mathbf{X}_d$	design at resolution d, $(\mathbf{X}_d)_{tB} = \mathbf{1}[B \subseteq S_t]$ for $\lvert B \rvert \le d$
$\mathbf{X}_1$	$[\mathbf{1}_T \mid \mathbf{X}]$, the design at $d = 1$
$\mathrm{freq}(m_i)$	$(\mathbf{X}^\top \mathbf{X})_{ii}$, the number of episodes that retrieve m_i
$\mathrm{co}(m_i, m_j)$	$(\mathbf{X}^\top \mathbf{X})_{ij}$, the number of episodes that retrieve both
$\mathbf{f}$	vector of retrieval frequencies, $f_i = \mathrm{freq}(m_i)$
$\mathbf{h}$	an invisible change, $\mathbf{h} \in \ker \mathbf{X}_d$
$\mathbf{e}_B$	unit vector for θ_B
$\mathbf{h}_B$	invisible change of a top-k log, Eq. Eq. (4)
s	scalar multiple of an invisible change
$\mathbf{w}$	least-squares solution of $\mathbf{X}\mathbf{w} = \mathbf{1}_T$
$n_b,\ T_b$	memories and episodes of block b
Estimation	
λ	ridge parameter, 10^{-2}
$\mathbf{V}$	$\mathbf{X}_d^\top \mathbf{X}_d + \lambda \mathbf{I}$
$\hat{\boldsymbol{\theta}},\ \hat{\varphi}_i$	ridge estimate of the dividends; estimate of φ_i, Eq. Eq. (5)
$\hat{\sigma}$	residual standard deviation
δ	error level of the intervals; 0.05 in Section 5
$z_{1-\delta/(2n)}$	Gaussian quantile of the Bonferroni interval
$\mathbf{U}$	orthonormal basis of $\ker \mathbf{X}_d$
τ	tolerance of the identifiability test, 10^{-8}
Merging	
$g,\ g(i)$	a group of memories; the group of memory i
$\bar{\mathbf{X}},\ \bar{\mathbf{X}}_1$	merged matrix, one column per group; with the baseline column
$\bar{\theta}_g$	value of group g, $\sum_{i \in g} \theta_{\{i\}}$
$s_g(\boldsymbol{\theta})$	$\sum_{i \in g} \theta_{\{i\}}$ for an arbitrary grouping (Theorem 3)
D_g	largest number of episodes in which group g was retrieved while one of its members was not
$\sigma_{\min}^+$	smallest nonzero singular value
Retrieval	
R_t	the M memories most similar to q_t
M	candidate window, 20
$f_i(t)$	number of episodes before t that retrieved memory i
$L_t,\ H_t$	least-retrieved and most similar parts of S_t, Eq. Eq. (8)
ρ	share of S_t filled by L_t, 0.4

Table 16: Terms, in the order they appear.

Episode	one query, one retrieval, one answer and its outcome
Log	the retrieved sets and outcomes of all episodes, stored as $\mathbf{X}$ and $\mathbf{r}$
Determined, identifiable	the log determines φ_i exactly when $\mathbf{c}^{(i)}$ lies in the row space of $\mathbf{X}_d$; the two words are used interchangeably
Undetermined	not determined; LEDGER reports no estimate for such a memory
Invisible change	a vector $\mathbf{h} \in \ker \mathbf{X}_d$: $\boldsymbol{\theta}$ and $\boldsymbol{\theta} + s\mathbf{h}$ predict the same outcome in every episode
Convention	a habit of retrieval or evaluation that fixes the counts of the log, such as a fixed size or a private store
Top-k log	a log in which every episode retrieves exactly k memories
Nested sets	sets retrieved for one query at several sizes, each containing the smaller ones, as under deterministic top-k
Private store	each question, document or conversation has its own memories
Shared store	one store used by all queries
Block	a group of memories never retrieved together with a memory outside it, with the episodes that retrieve them
Tied memories	memories with identical columns of $\mathbf{X}$, that is, retrieved in the same episodes
Merging	replacing each group of tied memories by one memory whose value is the group total
Unit	a memory or a group after merging; fractions after merging are over units
Fraction determined	share of memories, or units after merging, whose value the log determines
Pass	one round in which every question of the experiment is asked once
Configuration	one benchmark run at one retrieval size k in the audit
Gold memory	a memory that holds evidence for the answer
Gold recall	share of gold memories that top-k retrieves
Oracle	LEDGER applied only to questions where top-k misses a gold memory, which requires gold labels
Joint coverage	share of synthetic logs in which every reported interval contains its true value
Setting	one choice of n, k, T and λ in the synthetic study, with 20 random designs