WHAT DID THE AGENT LEARN FROM? CROSS-EPISODE MEMORY ATTRIBUTION FOR SELF-IMPROVING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic attribution identifies which component of an LLM agent system caused an observed success or failure. This supports debugging, accountability, and targeted improvement. Existing methods are largely within-episode: they assign outcomes to agents, actions, tool calls, or prompt context in the current trace. Persistent memory agents break this assumption because the relevant cause may not appear in that trace. A memory written many episodes earlier can affect an outcome directly through retrieval or indirectly through memories written later. We introduce memory-level attribution for self-improving agents and formalise these two paths as retrieval influence and generation influence. If no allowed intervention retrieves a memory, retrieval-only data may contain no evidence that distinguishes it from an absent memory. In our deployed retrieval configuration, individually deleting the never-retrieved memories leaves every measured top-k list unchanged. Retrieval-only data therefore do not record effects passed through later memories in these cases. This limitation matters at scale. At the deployed retrieval budget, 87–92% of a 278-memory pool is never retrieved. We therefore propose a visibility-aware attribution framework that matches the attribution unit to what can be observed. MEMATTR-V computes individual-memory Shapley values on the visible set, and MEMATTR-B computes lineage-band Shapley values on the observed shadow. Across three environments and four agent LLMs, MEMATTR-V's correlation with measured leave-one-in utility is positive in every evaluated cell, while MEMATTR-B's mean correlation is positive in every environment within task splits. We release the memory pools, generation lineages, and measured targets as MEMATTR-BENCH.

1 INTRODUCTION

Agentic attribution identifies which component of an LLM agent system caused an observed success or failure. It supports debugging, accountability, and targeted improvement. For an agent that accumulates and reuses memories, the cause may be a memory written many episodes earlier. Attribution must therefore extend beyond the failing episode. Deployed agent frameworks already support persistent memory (Packer et al., 2023; Chhikara et al., 2025). As deployment continues, the gap between the episode where a cause was written and the episode where its effect appears can keep growing. Inspecting only the failing trace then gives an incomplete account of what the agent learned and what should be changed.

Existing work attributes either multi-agent systems or individual agent traces. *Multi-agent attribution* asks which agent and which step in a cooperating team caused a task failure (Zhang et al., 2025). *Single-agent attribution* asks which step, tool call, or retrieved context item in one trace drove an answer (Cohen-Wang et al., 2024). Both families are *within-episode* despite attributing different units. They assign outcomes to components in the current trace and intervene by editing that trace. For memory, they observe only the items retrieved into the prompt, so they are *retrieval-only* (Figure 1). This design is sufficient when all causes of an outcome appear in the decision episode, but not for persistent agents. Persistent agents run across episodes and store lessons for later retrieval (Shinn et al., 2023; Wang et al., 2023; Park et al., 2023). This capability supports improvement, but it also lets one memory affect many later decisions. It can also propagate errors; Shao et al. (2026) observe

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

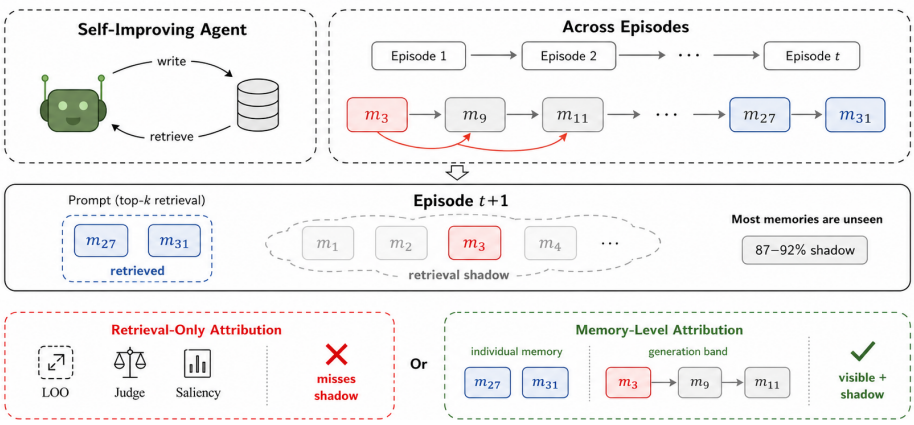

Figure 1: **Memory attribution across episodes.** Memory m_3 shapes later memories but is not retrieved in the decision episode. Blue nodes are retrieved memories; grey nodes are unretrieved memories. MEMATTR-V uses individual visible memories, and MEMATTR-B uses lineage bands. Values are illustrative.

safety-alignment degradation after memory accumulates. Cross-episode attribution therefore studies stored experience, including memories that shaped later context.

However, this persistent capability also creates three challenges for attribution (Figure 1). First, the causal memory can lie outside the episode being explained. A within-episode intervention assigns credit only among memories retrieved in that episode. Second, memories are dependent. An earlier memory can shape later memories, and deleting it after its descendants are written leaves those descendants unchanged. Weight-level unlearning does not remove these separate texts (Qu et al., 2025; Ding et al., 2025; Qian et al., 2026; Malki, 2026). An operator may delete the original memory yet leave its propagated content in the pool. Third, most memories are never retrieved during evaluation. A retrieval-only estimator cannot distinguish an unobserved memory from one with no measured effect. These cases require different actions, but they look identical in retrieval-only data.

We study memory-level attribution for self-improving agents. We separate direct retrieval effects from effects passed through later memories. We also use different attribution units depending on which memories an estimator can observe. Our contributions are:

- **Two cross-episode influence paths.** We formalise *retrieval influence*, transmitted when a memory is retrieved in the decision episode, and *generation influence*, transmitted through memories written later. This distinction separates direct retrieval from descendant-mediated effects and makes clear which counterfactual each quantity requires. In our 278-memory pool, the unretrieved portion reaches 87–92% at the deployed budget (§3).
- **A blind spot in retrieval-only attribution.** Under the deployed retriever, individually deleting each of 255 never-retrieved memories leaves all 7,905 measured top-3 lists unchanged. Retrieval-only data therefore provide no item-level evidence for these memories. In a separate case study, regenerating four memory lineages changes held-out success by up to 0.292, showing that the omitted write process can affect later behaviour in one pool (§4).
- **Visibility-aware estimators and MemAttr-Bench.** MEMATTR-V values individual retrieved memories. MEMATTR-B values *lineage bands*, groups descended from one source in the logged generation DAG, on the observed shadow. We evaluate both estimators against measured leave-one-in utility and release the pools, lineages, and targets as MemAttr-Bench (§5, §6.1). Within task splits, MEMATTR-V reaches mean $\rho = +0.79$ to $+0.86$, and MEMATTR-B has a positive mean ρ in every environment. Against granularity search on the shadow, MEMATTR-B has a mean paired within-split ρ difference of $+0.712$ (§6).

2 RELATED WORK

Agentic attribution. Failure attribution asks which agent, step, or tool call caused an outcome (Yao et al., 2023; Hong et al., 2024; Zhang et al., 2025; Cemri et al., 2025; Chen et al., 2026; Zhang

et al., 2026b). Context attribution asks which prompt item drove an answer (Lewis et al., 2020; Cohen-Wang et al., 2024; Chuang et al., 2025; Xie et al., 2024; Liang et al., 2025; Li et al., 2026c; Pan et al., 2026; Trinh et al., 2026). This task remains difficult within one episode: Zhang et al. (2026b) report less than 10% accuracy for step-level failure localisation by reasoning models. These methods intervene on a step, action, or context item in the episode that produced the outcome. They cannot attribute a stored memory that retrieval never surfaced. The difference is both the unit and the time window: we attribute stored memories whose effects may appear in later episodes.

Memory. Deployed agents build persistent pools with dedicated substrates and management policies (Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025; Yu et al., 2026; Jonelagadda et al., 2025; Sultan et al., 2026; Zhang et al., 2026a) and reuse distilled experience (Zhao et al., 2024; Wang et al., 2024). Existing benchmarks measure what a pool enables an agent to do (Hu et al., 2026; Cherif, 2026). Xiong et al. (2026) study memory addition and deletion, observe that errors propagate through stored experience, and use later task outcomes as quality labels. Such labels apply only to memories retrieved by the evaluated tasks. These studies measure the pool as a whole. They do not identify which memory helped, which memory hurt, or which earlier memory shaped the one that was used. We ask which memory affected an outcome and whether later memories carried its effect.

Self-improving agents. Agents that write and reuse their own experience (Shinn et al., 2023; Wang et al., 2023; Park et al., 2023) can improve or degrade through the same loop (Shao et al., 2026). Existing work evaluates whether persistent memory improves overall performance. It does not attribute outcomes to individual memories across episodes or measure indirect influence through later memory generation. Without that information, a system can remove a bad-looking memory while retaining later memories that copied its content. We study both questions and quantify how much of a growing pool remains observable to retrieval-based attribution.

3 Cross-episode memory attribution

Consider an agent that solves tasks drawn from a distribution while maintaining a pool M of self-written text memories. For a task τ, a retriever selects at most k memories, $S(\tau, M) \subseteq M$, and the agent acts on a prompt containing τ and $S(\tau, M)$. Let $\mu(\tau, M)$ denote the agent's success probability. When the agent writes a new memory, the memories retrieved at that time appear in its context. This induces a *generation DAG* $\mathcal{G}$, where an edge indicates that a memory was exposed during the generation of another memory. These are exposure edges, not causal ones: presence in context does not imply causal use. Reconstructing memory lineages therefore requires generation-lineage logs, which current agent frameworks rarely record.

We focus on persistent memory represented as discrete textual records and accessed through retrieval (Zhang et al., 2026a). This setting admits explicit memory-level interventions: records can be retrieved, removed, and linked across generations. We leave other forms of persistent state, such as online parameter updates and cached tool outputs, whose intervention structure differs from the memory-pool setting studied here, to future work. Within this scope, a memory can affect later outcomes either through direct retrieval or through subsequent memory writes, as formalized below.

Definition 3.1 (Two cross-episode influence channels)**.** For a memory m, let $M \setminus \{m\}$ denote the pool obtained by removing m while leaving all subsequently written memories unchanged. Let M^{-m} denote the counterfactual pool obtained by removing m and regenerating all subsequent memory writes, using the same tasks and random seeds. Then,

- The **retrieval influence** of m is $I_{\mathrm{ret}}(m) = \mathbb{E}_\tau\big[\mu(\tau, M) - \mu(\tau, M \setminus \{m\})\big]$, which measures the influence transmitted through the direct retrieval of m.
- The **generation influence** is $I_{\mathrm{gen}}(m) = \mathbb{E}_\tau\big[\mu(\tau, M) - \mu(\tau, M^{-m})\big]$, which also includes influence transmitted through memories generated from m.

Thus, $I_{\mathrm{gen}}(m) - I_{\mathrm{ret}}(m)$ is the effect passed through later memories.

The two quantities differ in what happens after m is removed. Retrieval influence keeps the realised memory pool fixed. Generation influence repeats later memory writes, so descendants may change or disappear. This matters when m affects the agent mainly through those descendants.

Table 1: Retrieval changes after single-memory deletion and held-out success after lineage regeneration. Panel (a) uses the 278-memory pool; Panel (b) uses the 20-memory ALFWorld `offset200` pool with baseline success 0.417.

(a) Retrieval deletion

k	shadow deletions	changed comparisons	visible controls
3	255	0/7,905	23/23
16	171	0/5,301	107/107

(b) Lineage regeneration

root	rewritten success	change from baseline
`fam0`	0.125	+0.292
`fam1`	0.500	−0.083
`fam2`	0.333	+0.083
`fam3`	0.250	+0.167

In practice, however, measuring I_{gen} requires regenerating subsequent memories, which is expensive and rarely supported. Most practical attribution procedures instead keep the realised memory pool fixed and rerun the agent under retrieval-side interventions. This raises an observability question: which memories can such procedures obtain evidence about?

Definition 3.2 (Retrieval-only estimator, and the two shadows)**.** Fix the sub-pool interventions on which an estimator may run the agent. An estimator is **retrieval-only** if its value depends only on the retrieved sets and outcomes observed under those interventions. The **observed shadow** contains memories that no task retrieved from the full pool; this is the retrieval shadow in Figure 1. The **unreachable shadow** contains memories that none of the allowed interventions can surface.

These two shadows distinguish memories missed by the observed retrieval from those that remain inaccessible under all allowed interventions, forming the observability gap studied next.

4 Limits of retrieval-only attribution

Retrieval-only methods, including leave-one-out, prompt saliency, and judges restricted to retrieved items, use only retrieval records and observed task outcomes. They therefore face two limitations: a memory may leave no item-level evidence under the estimator's retrieval interventions, and its effect may instead be transmitted through memories written later. Detailed setups for all measurements in this section are provided in the Appendix.

Unseen memories can leave no retrieval evidence. A memory in the observed shadow is not necessarily unreachable: although it is absent from the original retrieval results, removing it could still change which other memories are returned. We test this possibility by individually deleting every shadow memory and rerunning retrieval. In the tested pool, none of these deletions changes any measured top-k list, whereas deleting a visible memory always changes at least one list, shown in Table 1(a). Under these interventions, retrieval-only estimators therefore receive no item-level evidence about the shadow memories. This result is specific to the tested pool and retriever, and assumes that the agent accesses the memory pool only through retrieval (Appendix C.4).

Effects can propagate through later writes. Retrieval-side interventions also keep subsequently written memories fixed, and therefore do not capture effects transmitted through changes in those memories. We measure this missing path by removing a lineage root, regenerating its descendants, and reevaluating the resulting pool. Table 1(b) shows that lineage regeneration can substantially change held-out success, with the largest change reaching 0.292. Thus, an earlier memory can affect later behaviour through the write process even when ordinary fixed-pool attribution does not observe that path. This experiment is limited to one pool and four bands (Appendix D.7).

The unseen fraction grows with memory accumulation. The previous experiments show that retrieval-only attribution can miss both individual memories and effects carried through later writes. The importance of this limitation depends on how much of the memory pool retrieval actually exposes. Across P held-out episodes, a retriever with budget k returns at most kP memory slots, and repeated retrieval can concentrate those slots on the same memories as the pool grows. The shadow is not explained solely by a narrow evaluation workload: at a fixed goal count, drawing from more task types raises retrieval reach by 31–40%, yet the full audit still reaches only 23 unique memories from 93 available retrieval slots.

At k=3, 87–92% of the 278 memories from 12 interleaved sources are never retrieved (Table 2 and Figure 2). The range is similar across token overlap, two dense encoders (Reimers & Gurevych, 2019), and diversity-aware MMR (Carbonell & Goldstein, 1998); increasing the retrieval budget

Table 2: Never-retrieved fraction of a 278-memory agent-written pool under four retrievers and two retrieval budgets k, over P=31 held-out goals.

retrieval budget	token overlap	MiniLM	bge-small	MMR
$k = 3$ (deployed)	92%	87%	88%	88%
$k = 16$	62%	54%	59%	60%

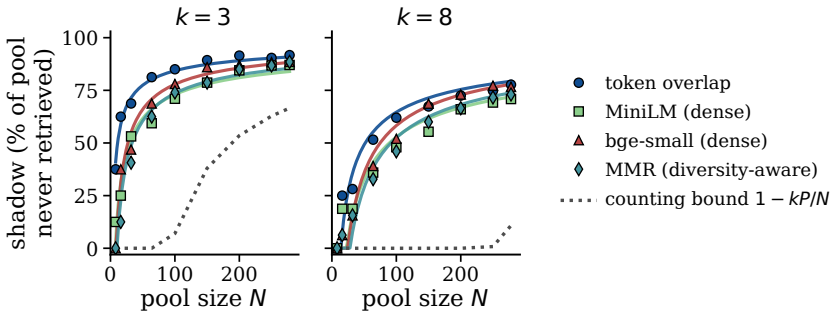

Figure 2: Retrieval coverage across pool sizes. Markers show measured never-retrieved fractions for up to 278 memories over P=31 held-out episodes. Curves show descriptive fits; the dotted line is the counting bound.

to k=16 still leaves 54–62% of the pool unseen. With the same P=31 held-out episodes at each pool size, retrieval exposes a decreasing share of the pool as the pool grows. Both power-law and logarithmic curves fit the measured range, and we do not extrapolate beyond it (Figure 2). The same pattern appears within a single agent run: the shadow rises from 38% to 79% while the task type remains fixed, showing that limited retrieval coverage persists even without increasing task heterogeneity.

These results motivate a different attribution unit for memories that lack item-level retrieval evidence. Generation lineage provides such a group-level unit, which we develop in Section 5.

5 Visibility-Aware Memory Attribution

In our single-memory deletion check, removing a never-retrieved memory does not change any measured top-k list (§4). Individual-memory attribution therefore lacks retrieval evidence for much of the observed shadow. We address this observability gap by matching the attribution unit to the evidence available in each part of the pool.

Changing the attribution unit. For the visible set, the unit is an **individual memory**. Removing that memory and rerunning retrieval measures its contribution to the fixed pool. This intervention can change which other memories are retrieved, so it captures the item's retrieval-side contribution across tasks. For the observed shadow, the unit is a **lineage band**: the memories descended from one source in the generation DAG. Removing a whole band creates a counterfactual even when none of its shadow members can be evaluated alone. Band attribution measures a group-level fixed-pool effect; it does not regenerate later memories. The coarser unit also reduces the number of players from 8–16 memories to 4–7 units in our evaluated pools (Appendix A.12). Figure 3 compares the individual-memory and lineage-band units used here.

Method 1: MemAttr-V for the visible set. MemAttr-V computes Shapley values over the pool-wide visible set. Each player is one memory retrieved by at least one derivation task, rather than the set retrieved on a single task. With k=3, our pools contain 4–6 visible players, so we enumerate every coalition. The Shapley enumeration is exact, while each coalition value is estimated from a finite number of agent rollouts (Algorithm 1). This uses the standard data-Shapley objective (Ghorbani & Zou, 2019), but evaluates in-context memories without retraining. In particular, the method evaluates how the agent behaves under each sub-pool; it does not infer value from the memory text alone. The player set is the same for every held-out task. This makes the resulting values directly comparable across held-out tasks.

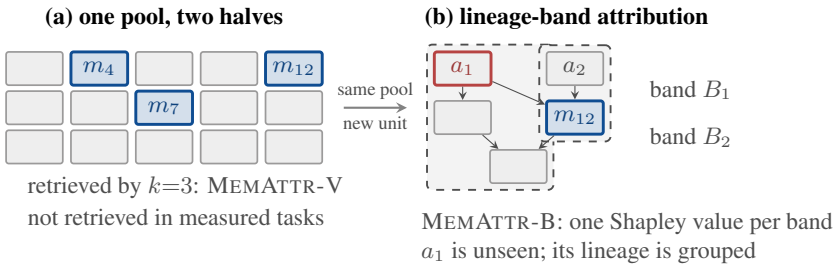

Figure 3: Two attribution units. (a) Retrieval divides the pool into the visible set (blue) and observed shadow (grey). (b) Ancestry in the generation DAG groups memories into disjoint lineage bands.

Table 3: MemAttr-Bench composition. All memories are agent-written. $v(\emptyset)$ is the measured success with an empty memory pool, which provides the reference level for attributed contributions.

environment	pool size	$v(\emptyset)$	agent LLM
ALFWorld	16	0.190	Ministral-14B / Qwen3-32B / Llama3.1-8B
ScienceWorld	8	0.500	Qwen3-32B
τ-airline	12	0.500	Sonnet-4.5
τ-retail	12	0.375	Sonnet-4.5
retrieval-shadow pool	278	n/a	12 sources, interleaved

Method 2: MEMATTR-B for the shadow. MEMATTR-B computes Shapley values with lineage bands as players. The reported bands use source ancestry in the logged generation DAG. A source is a memory written with no retrieved memories in context, and each descendant is assigned to its source ancestor. Every evaluated memory has exactly one source ancestor, so no edge weight enters the reported partition. Withholding a band removes its already-written memories from the evaluation pool without rerunning memory generation. MEMATTR-B therefore measures the fixed-pool value of a realised lineage, rather than the per-memory generation influence I_{gen} in Def. 3.1. The regeneration experiment in Section 4 illustrates this distinction.

We estimate band Shapley values with permutation Monte-Carlo (Castro et al., 2009) and compare with exhaustive band Shapley where feasible. Each sampled permutation measures the marginal value of adding a band after the bands that precede it. Coalition values are cached and reused across permutations. More permutations reduce sampling error, but they do not average away noise in a cached coalition value. In our cached-noise simulation, the error flattens after about 20 permutations; more rollouts per coalition are then more useful than more permutations.

6 EXPERIMENTS

Section 4 examines two ways to delete a memory: keeping later memories fixed or regenerating them (Definition 3.1). Following the two-unit design in Section 5, we evaluate MEMATTR-V on visible memories and MEMATTR-B on lineage bands in the observed shadow. Both methods score the written pool and do not estimate generation influence. Leave-one-in utility is success gained by adding a memory or band to an empty pool.

6.1 EXPERIMENTAL SETUP

MemAttr-Bench. We release the agent-written memory pool for each environment, its **generation DAG**, and measured leave-one-in utility for each memory and band. The DAG records which memories were in context when each later memory was written. It is required to reconstruct the lineage bands used for band-level attribution, but current agent frameworks generally do not record this information. The benchmark therefore contains both the final memory pool used for evaluation and the generation lineage needed to reproduce the band partition. Appendix C.1 gives the retrieval settings and derivation/evaluation split protocol.

Baselines. Existing attribution methods primarily operate on steps or context within a single trace, so we adapt five baselines to the memory setting. **M1**, the *whole-pool judge*, asks an LLM to rank memories from their text. It can score memories that retrieval never returns, but it does not rerun

Table 4: Within-split Spearman ρ against measured leave-one-in utility. Superscript n gives the number of pool–split evaluations and the subscript gives the observed range. The top block uses the same retrieved-memory unit; methods in the bottom block are evaluated on their respective attribution units.

method	ALFWorld	SciWorld	τ-bench
visible set, per retrieved memory			
M2 item LOO	$+0.15^{(6)}_{[-0.02,\,0.27]}$	$+0.03^{(2)}_{[-0.08,\,0.14]}$	$-0.15^{(2)}_{[-0.95,\,0.65]}$
MEMATTR-V	$\mathbf{+0.86}^{(6)}_{[0.67,\,0.98]}$	$\mathbf{+0.80}^{(3)}_{[0.58,\,1.0]}$	$\mathbf{+0.79}^{(2)}_{[0.67,\,0.90]}$
reaches the observed shadow; each method evaluated on its own unit			
M1 whole-pool judge	$+0.23^{(2)}_{[0.19,\,0.26]}$	$+0.38^{(2)}_{[-0.07,\,0.84]}$	$-0.46^{(1)}$
M4 granularity search	$+0.14^{(5)}_{[-0.50,\,0.83]}$	$-0.97^{(2)}_{[-1.0,-0.94]}$	—
MEMATTR-B	$+0.72^{(6)}_{[0.20,\,1.0]}$	$+0.45^{(2)}_{[0.26,\,0.63]}$	$+0.50^{(1)}$

the agent after changing the pool. **M2**, *item leave-one-out*, removes one memory, reruns retrieval, and measures the resulting change in success. **M3** walks the logged generation DAG from a failure toward its ancestors and returns a candidate source. **M4**, *granularity search*, partitions the pool into 1, 2, and 4 token-overlap clusters, scores each part on two disjoint task splits with 4 seeds, and returns the finest level whose contribution keeps the same sign across splits. **M5**, *binary search*, repeatedly retains the half whose removal changes the outcome until one memory remains. M3 and M5 return a single source rather than a ranking, so they are excluded from the rank-correlation comparison in Table 4.

Environments and protocol. We evaluate on ALFWorld (Shridhar et al., 2021), ScienceWorld (Wang et al., 2022), and both domains of τ-bench (Yao et al., 2024). Four agent LLMs write pools of 8–16 memories (Table 3).

For ScienceWorld, we screened pools on 12 task variations by requiring the full pool to outperform an empty prompt. On a disjoint set of task variations, the mean difference fell from $+0.200$ to $+0.017$. We therefore use the original result only to select pools, not as evidence that a retained pool improves performance. In half of the pool–split evaluations, the full set of bands does not outperform the empty pool. We therefore also report the band-level average for the evaluations where it does. Appendices D.13 and D.1 give the corresponding analyses.

We use three reporting conventions. First, positive influence I means that a memory improves success, while *blame* is $-I$. Second, our utility metric is measured *leave-one-in utility*: for a memory or band, it is the change in held-out success when that unit is added to an empty memory pool. This is an evaluation target, not I_{ret} or I_{gen}. Agreement means rank correlation with this measured utility. A Shapley *player* is the unit being scored: one visible memory for MEMATTR-V, or one lineage band for MEMATTR-B. Third, every p-value uses tasks, rather than (task, seed) cells, as the statistical unit. At temperature 0, treating seeds as independent observations would inflate the effective sample size. The attribution experiments use 31,957 agent episodes in total (Appendix D.14).

6.2 MAIN COMPARISON AGAINST THE BASELINES

The methods operate on different attribution units. M2 and MEMATTR-V assign values to individual memories exposed by retrieval. M1, M4, and MEMATTR-B can reach the observed shadow, but use different information or attribution units. Table 4 therefore separates the item-level comparison from the shadow-reaching methods. Only M2 and MEMATTR-V score the same memories. The bottom block reports each method on its own unit and is not a direct method comparison.

Visible-set attribution. Item leave-one-out is inconsistent across environments. Its correlation ranges from $+0.65$ on τ-airline to -0.95 on τ-retail, although the two domains use the same benchmark, protocol, and evaluation target. It also produces only 2–3 distinct values across 5 memories. In contrast, MEMATTR-V has positive correlation in every evaluated environment, although it applies only to memories exposed by retrieval. Within the tested splits, these results are consistent with using individual-memory attribution where retrieval provides item-level evidence.

Leave-one-in utility is already one component of each Shapley value, while item leave-one-out evaluates removal from the full pool. We therefore repeat the comparison after removing this shared component. The resulting estimator has mean correlation $\rho{=}+0.57$, compared with $+0.28$ for item

Table 5: Band Shapley and M4 on six band pools (12 pool–split evaluations), measured by Spearman ρ against leave-one-band-in utility. *exact* is exhaustive Shapley and *perm-MC* is permutation Monte Carlo. *TMC* is truncated Monte Carlo; = denotes the same ranking as perm-MC, and *n/a* means that the full band set has no positive value to truncate against. ‡ marks a pool–split evaluation with no comparable M4 result. Bold marks the larger perm-MC or M4 value.

	split a				split b			
pool	**exact**	**perm-MC**	**TMC**	**M4**	**exact**	**perm-MC**	**TMC**	**M4**
ALFWorld-Ministral	0.77	0.77	=	0.77	1.00	**1.00**	=	0.00
ALFWorld-Qwen	0.20	**0.20**	=	−0.40	0.63	0.63	=	**0.83**
ALFWorld-Llama	0.83	**0.95**	n/a	−0.50	0.82	0.77	n/a	—‡
ALFWorld-branched	0.60	**0.60**	=	−0.21	−0.21	**−0.21**	=	−0.89
ALFWorld-offset200	0.32	**0.32**	n/a	−0.78	−0.82	−0.77	=	**−0.33**
ScienceWorld	0.82	**0.63**	=	−0.94	0.00	**0.26**	n/a	−1.00

Table 6: Blame in one logged lineage. *ret*: measured $-I_{\mathrm{ret}}$; *ind*: blame passed back from descendants. Positive means harm; *ind* is not a Shapley value.

memory	**ret**	**ind**	**role**
`97e2d6d6`	+0.167	0.000	direct cause
`48aea935`	−0.333	0.000	innocent **indirectly implicated**
`5cb6b7fe`	0.000	+0.133	never retrieved

leave-one-out. Five of seven pool rankings are unchanged, while τ-retail changes to -0.40 (Appendix D.9). The shared component therefore explains part, but not all, of the observed difference.

Band-level attribution on the observed shadow. MEMATTR-B is positive in every environment in Table 4, although its ranges are wider. Granularity search is negative on average in ScienceWorld. Across six band pools and two task splits, band Shapley has higher correlation than M4 in 8 of 12 pool–split evaluations, with 1 tie, 2 losses, and 1 evaluation without a comparable M4 result (Table 5). Band Shapley is negative in 2 evaluations, compared with 8 for M4. The win count is not significant (sign test p=0.11), while the mean difference within the same pool–split evaluation is $+0.712$, with bootstrap confidence interval $[+0.315, +1.069]$.

The cross-split result is also inconclusive at this sample size. When attribution is derived on one split and evaluated against the measured target on another, the band-level test gives p=0.274. We therefore treat the band results as within-split evidence rather than evidence that the ranking transfers to another task split (Appendix D.16).

Lineage grouping itself does not show an accuracy advantage over random grouping in the control we tested. On one 20-memory pool, the lineage-based four-band partition reaches ρ=+0.600, while three size-matched random partitions average $+0.630$. We use lineage because it records where the memories came from and supports regenerating descendants, not because this experiment shows higher ranking accuracy than random grouping (Appendix D.3). Permutation Monte Carlo closely matches exhaustive band Shapley on the evaluated pools; Appendix A.7 studies approximation error and scaling to larger band counts.

Robustness across agent LLMs. Across ALFWorld pools written by three agent LLMs, both MEMATTR estimators remain positively correlated with their measured targets, although the magnitudes vary; Appendix D.15 reports the full cross-model diagnostics. As a sanity check, both baselines also recover the single decisive memory in a controlled six-memory setting (Appendix A.6).

A worked lineage example. The aggregate comparisons above measure ranking quality, but they do not show how an earlier memory can be implicated through a later write. Table 6 reports two scores for one logged lineage: retrieval blame is the measured leave-one-out quantity $-I_{\mathrm{ret}}$ in Definition 3.1, while the indirect score passes part of a child's blame to its logged ancestors. The directly retrieved harmful memory receives positive retrieval blame, while the never-retrieved ancestor receives zero retrieval blame but a nonzero indirect score. A second retrieved memory has the largest-magnitude score, but its negative sign indicates that removing it would hurt the agent. The two scores should not be collapsed into one ranking.

Taken together, the two estimators operate at different levels of observability. MEMATTR-V assigns individual values where retrieval provides item-level evidence, while MEMATTR-B assigns group-

Table 7: One-memory pruning on τ-retail. P0 is success before deletion. P3 deletes the negative-utility memory; P5 deletes one at random. Other entries are changes from P0.

	split-a	**split-b**
P0 keep everything	$56/80 = 0.700$	$59/80 = 0.738$
P3 prune negative-utility memory	-0.075	-0.050
P5 prune same number at random	$+0.062$	$+0.000$
task-level P3 vs. P5	P3 wins 0, P5 wins 7	P3 wins 1, P5 wins 4
no-deletion control	$+0.000$	$+0.013$

level values to realised lineages that extend into the observed shadow. The indirect score in Table 6 is a heuristic lineage score rather than I_{gen}. Neither estimator regenerates the written memory pool, so neither recovers per-memory generation influence. Measuring I_{gen} still requires regenerating subsequent memories.

6.3 Downstream task: target-guided pruning

Deleting the negative-utility memory does not beat random deletion. Target-guided means selecting the memory by measured leave-one-in utility, not by MemAttr-V or MemAttr-B. We compare this deletion with deleting one random memory on 40 held-out τ-retail tasks not used elsewhere. The pre-registered one-sided task-level test finds no evidence that the negative-utility deletion is better (p=0.9998); both split estimates instead favour random deletion (Table 7). The test can detect only large effects, so we do not claim that random deletion is better (Appendix D.18).

All item- and band-level estimates on this pool are positive, so neither estimator selects a memory for deletion. The experiment also removes one memory, whereas MemAttr-B scores lineage bands. It therefore tests neither a deletion chosen by our estimators nor deletion of a whole band.

Repeating the same configuration without deletion measures run-to-run variation. This no-deletion control changes success by $+0.054$ on ALFWorld, more than any pruning effect measured there. On τ-retail, the split-level changes are $+0.000$ and $+0.013$, although 16–18% of task–seed results still change between identical runs. We therefore use paired task-level tests and measure this variation separately in each environment.

The benchmark asks whether rankings agree with leave-one-in utility; pruning asks what a specific deletion does. The first result does not imply the second. We therefore treat attribution as a diagnosis, not a deletion rule. Any deletion policy must choose between memories and bands, compare with both random deletion and no deletion, and report ranking quality separately from the effect of the deletion.

7 Conclusion

We introduced memory-level attribution for self-improving agents. The framework separates retrieval influence from generation influence across episodes. Retrieval-invariance, retrieval-coverage, and direct-regeneration experiments show that retrieval logs omit many memories and do not record all effects passed through later memory writes. We therefore compute individual-memory Shapley values on the visible set and fixed-pool lineage-band Shapley values on the observed shadow. These estimates answer different questions, so we report them separately. MemAttr-V's correlation with measured leave-one-in utility is positive in every evaluated cell, while MemAttr-B's mean correlation is positive in every environment within task splits. We release the memory pools, generation lineages, measured targets, and evaluation protocol as MemAttr-Bench. A pre-registered single-memory deletion test did not outperform random deletion. This result separates single-memory target-guided pruning from lineage-aware editing, which remains open. A system that supports lineage-aware attribution must record which memories were available when later memories were written. The next steps are to log this lineage, test band-level editing with null-arm calibration, and extend cross-episode attribution to other forms of persistent agent state. More broadly, the attribution unit should match a counterfactual that the system can execute.

AI USE STATEMENT

Required-disclosure uses. We used generative AI tools to review the experimental methodology. This review corrected the statistical unit from the (task, seed) cell to the task and helped design the pre-registered replication in §6.3. We also used the tools to implement the methods and interpret results. We checked all claims and take responsibility for the modelling choices and conclusions.

We did *not* use them to generate data standing in for measured results, to clean or reformat data, for translation, or for qualitative analysis. No result, measurement, or citation here was produced by a model describing a run: every reported quantity is the output of executed code on the released artefacts. The memory pools of MemAttr-Bench are LLM-written by construction, since agent-written memory is the object of study; that is data collected from agents, not authoring assistance.

Recommended-disclosure uses. We used them to write and debug the experiment code, to orchestrate the runs, and to draft and edit the manuscript prose.

We have reviewed all AI-assisted work and take responsibility for the final content of this paper.

ETHICS STATEMENT

This work involves no human subjects or personal data. Every memory is text that an agent wrote about its own episodes in a public simulated benchmark: ALFWorld, ScienceWorld, τ-bench, or SWE-bench-Lite. The released artefacts inherit the terms of these benchmarks.

A relevant misuse risk is automated memory deletion: if a method names the memory responsible for a failure, the apparent next step is to remove it. Our results do not support automated deletion based on attribution alone. Target-guided one-memory deletion did not beat size-matched random deletion, and the point estimates favoured the random control (§6.3). A text-only LLM judge's ranking is a fixed function of memory text and does not depend on the task distribution. The same ranking scores $+0.84$ and -0.07 on two disjoint splits of one pool (Appendix D.10.3). Because an LLM judge is inexpensive, it is a plausible choice for memory curation. Its ranking still needs validation on the target task distribution. We recommend the null-arm calibration of Appendix C.6 before any automated memory-editing system is deployed.

Finally, on cost: the experiments consumed $\approx 32{,}000$ agent episodes, and the single most expensive leg returned a null. We report the per-leg cost (Appendix D.14) so that others can weigh it before repeating the experiment.

REPRODUCIBILITY STATEMENT

All quantities in this paper are measured, not annotated. The release will be organised so that each quantity can be recomputed. MemAttr-Bench is not available during review, which limits verification of the benchmark contribution at submission time.

Every number is reported with its unit and player set. The release will contain the agent-written memory pools, the generation DAG for each pool, the measured leave-one-in and leave-one-band-in targets, the retrieval configurations, and the derivation/evaluation split protocol. The retrieval configurations specify the retriever, k, and the interleaving used to build the 278-memory pool.

Section 4 uses the retrieval-invariance check in Appendix C.4, the retrieval-coverage measurements in Figure 2 and Table 2, and the regeneration experiment in Appendix D.7. Appendix A gives the supporting definitions and analyses. For §6.3, we include the pre-registration verbatim. The hypothesis, analysis unit, test, task ids, and decision rule were fixed before the confirmatory data existed. We report the outcome returned by that rule.

REFERENCES

Anthropic. Claude sonnet 4.5. `https://www.anthropic.com/news/claude-sonnet-4-5`, 2025.

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent LLM systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

Zhi Chen, Wei Ma, and Lingxiao Jiang. Beyond final code: A process-oriented error analysis of software development agents in real-world GitHub scenarios. In *International Conference on Software Engineering (ICSE)*, 2026.

A. Cherif. AgentMemBench: A systematic benchmark for evaluating long-term memory management strategies in conversational AI agents. *arXiv preprint arXiv:2608.00009*, 2026.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *European Conference on Artificial Intelligence (ECAI)*, 2025.

Yung-Sung Chuang, Benjamin Cohen-Wang, Zejiang Shen, Zhaofeng Wu, Hu Xu, Xi Victoria Lin, James Glass, Shang-Wen Li, and Wen-tau Yih. SelfCite: Self-supervised alignment for context attribution in large language models. In *International Conference on Machine Learning (ICML)*, 2025.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. ContextCite: Attributing model generation to context. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. TextWorld: A learning environment for text-based games. In *Computer Games Workshop at IJCAI*, 2018.

Meng Ding, Rohan Sharma, Changyou Chen, Jinhui Xu, and Kaiyi Ji. Understanding fine-tuning in approximate unlearning: A theoretical perspective. *Transactions on Machine Learning Research (TMLR)*, 2025.

Meng Ding, Zeqing Zhang, Di Wang, and Lijie Hu. In-run data shapley for adam optimizer. *arXiv preprint arXiv:2602.00329*, 2026.

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

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations (ICLR)*, 2024.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.

Aneesh Jonelagadda, Christina Hahn, Haoze Zheng, and Salvatore Penachio. Mnemosyne: An unsupervised, human-inspired long-term memory architecture. *arXiv preprint arXiv:2510.08601*, 2025.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017.

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.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. Preference leakage: A contamination problem in LLM-as-a-judge. In *International Conference on Learning Representations (ICLR)*, 2026a.

Guangyu Li, Meng Ding, and Lijie Hu. Understanding generalization and forgetting in in-context continual learning. In *International Conference on Machine Learning (ICML)*, 2026b.

Ruizhe Li, Chen Chen, Yuchen Hu, Yanjun Gao, Xi Wang, and Emine Yilmaz. Attributing response to context: A jensen-shannon divergence driven mechanistic study of context attribution in retrieval-augmented generation. In *International Conference on Learning Representations (ICLR)*, 2026c.

Zheng Liang, Hongzhi Wang, Xiaoou Ding, Zhiyu Liang, Cheng Liang, Yafeng Tang, and Jianzhong Qi. Tailoring the shapley value for in-context example selection towards data wrangling. In *IEEE International Conference on Data Engineering (ICDE)*, 2025.

Llama Team, AI @ Meta. The Llama 3 herd of models. Technical report, Meta, 2024.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Lisa Mekioussa Malki. Towards usable, privacy respecting long-term memory for LLM-based conversational agents. In *CHI Conference on Human Factors in Computing Systems, Extended Abstracts*, 2026.

Guillermo Owen. Values of games with a priori unions. *Mathematical Economics and Game Theory*, pp. 76–88, 1977.

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

Deng Pan, Keerthiram Murugesan, Ting Hua, Nuno Moniz, and Nitesh V. Chawla. Context attribution with multi-armed bandit optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 11651–11662, 2026.

Joon Sung Park, Joseph O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

Wei Qian, Chenxu Zhao, Yangyi Li, and Mengdi Huai. Towards benchmarking privacy vulnerabilities in selective forgetting with large language models. In *AAAI Conference on Artificial Intelligence*, 2026.

Youyang Qu, Ming Ding, Nan Sun, Kanchana Thilakarathna, Tianqing Zhu, and Dusit Niyato. The frontier of data erasure: A survey on machine unlearning for large language models. *Computer*, 2025.

Qwen Team. Qwen3 technical report. Technical report, Alibaba Group, 2025.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Chenyang Ren, Huanyi Xie, Shu Yang, Meng Ding, Lijie Hu, and Di Wang. Evaluating data influence in meta learning. arXiv:2501.15963, 2025.

Shuai Shao, Qihan Ren, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, Dongrui Liu, and Jing Shao. Your agent may misevolve: Emergent risks in self-evolving LLM agents. In *International Conference on Learning Representations (ICLR)*, 2026.

Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2:307–317, 1953.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

M. Sultan, Sri Thuraisamy, and D. Rajaratnam. NeuSymMS: A hybrid neuro-symbolic memory system for persistent, self-curating LLM agents. *arXiv preprint arXiv:2605.17596*, 2026.

Quoc-Huy Trinh, Lin Zhu, and Sebastian Szyller. How context attribution handles what the model already knows. *arXiv preprint arXiv:2607.23804*, 2026.

Dmytro Vitel and Anshuman Chhabra. First is not really better than last: Evaluating layer choice and aggregation strategies in language model data influence estimation. In *International Conference on Learning Representations (ICLR)*, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Shan Xie, Man Luo, Chadly Daniel Stern, Mengnan Du, and Lu Cheng. DemoShapley: Valuation of demonstrations for in-context learning. In *IEEE International Conference on Big Data*, 2024.

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

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, 2026.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents. In *International Conference on Learning Representations (ICLR)*, 2026a.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems? In *International Conference on Learning Representations (ICLR)*, 2026b.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599. PMLR, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *AAAI Conference on Artificial Intelligence*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A Definitions and supporting analysis

A.1 Scope and counting limit

Scope. Generation-DAG edges record which memories were in context when a later memory was written. They do not show that the write process used those memories. This is why the optional rule in Appendix A.5 uses a stance gate instead of treating every logged edge as causal. Our scope is a pool of discrete texts surfaced by a retriever, not generative latent memory (Zhang et al., 2026a), which has no retrieved set to ablate.

Our regeneration experiment reuses the same tasks and random seeds after a memory is removed. It measures how later memory writes change on one pool (Appendix D.7).

Counting limit for retrieval coverage. A deterministic retriever returns at most k memories per task. Across P distinct retrieval profiles, it can surface at most kP distinct memories. The rest of an N-memory pool remains in the observed shadow. A fixed audit set therefore covers a smaller share as the pool grows.

A.2 Intervention families for retrieval-only estimators

Which estimators are in the class. The class is broad: leave-one-out, single-point deletion, Shapley over the retrieved set, prompt-side saliency and a judge shown *the retrieved list* are all retrieval-only for their own intervention sets, while the whole-pool judge is deliberately *not*, which is why it is the one baseline that can name a shadow memory.

If an estimator could test every possible sub-pool, a top-k retriever would surface any memory when that memory is tested alone. The relevant intervention family is therefore the one the estimator actually uses. Item leave-one-out tests single deletions, prompt-side estimators use only the full pool, and coalition methods use the sub-pools they enumerate. Our measured fraction is the never-retrieved set on the sampled tasks (Appendix C.3).

A.3 SPLIT DEPENDENCE AND NON-ADDITIVITY

Results on the visible set. An item's leave-one-out value depends on the task distribution and can change sign across splits. Across three split axes in two environments, the harmful-memory sets overlap by only 0%–21%. Summed leave-one-out credit ranges from 0% to 300% of the measured pool effect where a full coalition table is available. In contrast, Shapley values over the same items sum to the pool effect by construction. The split dependence and non-additivity reported here therefore apply to flat leave-one-out, not to item-level Shapley.

Why zero-overlap cells occur. In two cells, one split has no harmful set. At item level, split a finds no harmful memory while split b finds five. At group level, split a finds three harmful groups while split b finds none. The harmful sets therefore have zero overlap. Thus, the sign of a small-unit effect does not survive a split change. The result does not compare two populated sets. If both sets are empty, the overlap is undefined rather than zero; that case does not occur here.

Non-additivity and cross-split ranking. In one pool, summed item credit is $+0.278$ while the measured pool effect is $+0.167$. The item credits therefore exceed the pool effect by $+0.111$ and reach 167% of it because the removals are redundant. The consequence affects the ranking: attribution derived on one split can give the memory that is *most* valuable on a disjoint split a *negative* value (-0.049 against a measured $+0.381$, rank 15 of 16). And the instability is a property of the *granularity*: the measured-best unit survives a change of split on 1 of 3 pools at item level but 4 of 6 at band level, and on the one pool partitioned both ways it is stable at four bands and unstable at seven (Appendix D.16). The difference is not statistically significant ($4/6$ against $1/3$ is Fisher exact two-sided $p = 0.52$; $4/6$ alone is $p = 0.34$ against chance, over 6 pools). We therefore treat it as a directional observation rather than evidence of improved stability. Together with the measured retrieval coverage in §4, the result has three parts. Item-level attribution has no player for most of the pool, is split-unstable on the visible set, and is not a decomposition of the total effect. This is a property of the estimator and task distribution, not of the attribution question itself. An item-level effect that changes across sub-distributions is distribution-dependent, not ill-defined. At this sample size, flat LOO does not recover a distribution-stable ordering.

The non-additivity result appears in multiple pools. We recomputed the additivity gap wherever a full coalition table permits. At band level, we remove one band from the full band set, and every required value is already cached. The summed leave-one-out credit is 300% of the measured pool effect on `dcfine_a` ($+0.857$ against $+0.286$) and 0% on `dcfine_b` ($+0.000$ against $+0.250$), the latter is a case of perfect redundancy: *no single band* changes success when removed alone, while removing all of them reduces success by 0.25, so leave-one-out assigns zero total credit to a pool with a positive measured effect. On the ScienceWorld pool the pool effect is exactly 0 and the ratio is undefined, though the summed credit is $+0.167$. The defect is two-sided: flat LOO over-credits by threefold on one pool and under-credits to nothing on another, which is why we treat it as non-additivity rather than as a bias with a sign we could correct for.

Our measurements separate three granularities. At pool level, the effect is sign-stable across all six splits in both environments. At generation-band level, a whole band can be removed and evaluated. Small retrieval-defined units, including single memories and topic groups of four, are split-unstable, with no overlap between harmful groups in the measured group-level comparison. The rest of the paper therefore makes pool- or band-level claims and reports item-level values only for a stated task distribution.

A.4 STABILITY METRICS AND EVALUATION CONVENTIONS

Sign convention. Every I in Def. 3.1 is a *value*: positive means the memory helped, because deleting it lowered success. Every quantity we call *blame* is its negation, so positive blame means the memory hurt. We flag the convention here because both readings are natural and the two appear in this paper: rankings and correlations use I, while the role assignment of §6.2 and the lineage propagation of Appendix B.2 are reported as blame.

Evaluation target. For a memory or band, leave-one-in is its success when inserted alone minus empty-pool success. LOO is a *method* we evaluate (M2), not a target: LOO removes and LOI inserts, and on interacting memories the two need not agree.

LOI is a *stand-alone utility*, not I_{ret}, I_{gen}, or the Shapley value we compute. We therefore call it an **evaluation target**. Agreement with it does not establish recovery of I_{gen}. We score against it because it is measured rather than annotated, defined even for a never-retrieved unit, and common to every method. However, leave-one-in is also the singleton term in the Shapley average, so estimator and target share a term. Appendix D.9 tests this overlap by repeating every correlation with that term removed.

Stability metrics. Split-stability is the Jaccard overlap between the harmful-memory sets found on two task splits. The additivity gap is the summed item credit minus the measured effect of the whole pool. All regeneration runs reuse the same tasks and random seeds.

A.5 Band construction

Stance-gated lineage weight. For each parent–child edge, an entailment judge labels whether the parent supports, contradicts, or is unrelated to the claim extracted from the child alone. Supporting parents receive normalised embedding-similarity weights; other parents receive zero. An ancestor's weight on a descendant sums the products of edge weights along all directed paths between them. If no parent supports the child, the child becomes a new source. Claim extraction and entailment each require one LLM call per edge; both prompts are in Appendix B.3.

Band partition. The sources are memories written with no retrieved context. Since the generation DAG is acyclic, every memory has a source ancestor. We assign each memory to the source with the largest cumulative lineage weight and break ties in favour of the earliest-written source. This hard assignment creates disjoint lineage bundles.

In our evaluated pools, every memory has at most one source ancestor ($0/20$ on `branched`, $0/20$ on `offset200`, $0/12$ on the ScienceWorld pool). The weighting choice therefore does not affect the reported partitions. We use the source-rooted partition of the logged DAG and use the weights only to describe provenance within a band. Appendix A.9 applies the stance gate before assignment and obtains a different partition. MemAttr-B treats the resulting bands as disjoint Shapley players.

Band influence. Band influence is the change in expected success after every memory in a band is removed and none is regenerated. MemAttr-B treats each band as a player and returns its standard Shapley value from the measured coalition outcomes. This quantity is not the generation influence of the band's source. The band intervention deletes the descendants instead of regenerating them, so it charges the source for the texts its descendants currently hold rather than for the difference between those texts and the ones a re-run would produce. The two agree when re-running generation without the source writes nothing in the descendants' place, and they can differ in either direction otherwise: a source whose descendants would have been written anyway is over-charged, and one whose absence would have produced worse descendants is under-charged. Appendix D.7 measures the gap on the one pool where we re-ran generation, where band Shapley tracks measured I_{gen} at $\rho{=}+0.32$ over four bands.

A.6 Constructed control with 87 verified instances

The cross-split nulls of §6.2 may reflect weak estimators or unstable per-memory targets in natural pools. This control constructs instances with a verified decisive memory and applies the same estimators.

Construction. Each instance is one ALFWorld task with six memories: one *decisive* lesson written for that task's layout, and five distractors drawn from other layouts and task types. Decisiveness is not asserted, it is **verified by rollout**: we keep an instance only if the agent's success rate with the decisive memory present exceeds its rate with the memory removed, and we record both together with the empty-pool rate. Of 280 candidates over five task types, 87 passed and 193 were rejected.

The retained instances have a mean success gap of 0.86 with a minimum of 0.25, drawn from 28 evaluation and 59 training layouts.

Result. Scored by whether the estimator ranks the decisive memory first:

task type	n	M1 whole-pool judge	M2 item LOO
pick_and_place	22	0.95	0.91
pick_clean	15	1.00	0.73
pick_cool	13	1.00	0.85
pick_heat	8	1.00	0.62
pick_two	29	1.00	0.66
all	87	**0.989** (86/87)	**0.759** (66/87)

Chance is $1/6$, so both are far from it ($p = 8.7 \times 10^{-66}$ and 7.7×10^{-34}, exact binomial). MRR is 0.994 and 0.852.

Scope of the constructed control. On these constructed pools, the judge ranks the decisive memory first in 86 of 87 cases and item LOO in 66 of 87. This shows that the implementations can recover a strong, verified single-memory signal. It does not make 0.989 a performance estimate for natural pools. The construction removes interactions and target ambiguity, so it provides an upper-bound control.

A.7 Permutation Monte Carlo accuracy and scaling

With four bands, the permutation estimator is exhaustive, so Table 5 validates band-level Shapley against granularity search but says nothing about the approximation. Three measurements answer that, all without new rollouts.

Real-data approximation error. The two dcfine pools have seven players and complete tables for all 128 coalitions, so exact Shapley is available. Averaged over 20 repetitions, RMSE against exact falls from 0.080 with two sampled permutations to 0.006 with 400 on dcfine_a; the intermediate values are reported below, and dcfine_b stays within 0.006 of them. With 400 permutations, permutation error is a fifth of the smallest band effect we report.

Effect of cached rollout noise. We perturb each coalition value by the noise of eight Bernoulli rollouts. We then compare reusing that value across permutations, as our cache does, with redrawing it for every permutation on dcfine_a:

permutations	2	5	10	19	50	100	200	400
cached (ours)	0.184	0.125	0.096	0.081	0.067	0.060	0.057	0.056
fresh draws	0.179	0.113	0.081	0.059	0.035	0.023	0.017	0.012
ratio	1.03	1.11	1.18	1.38	1.93	2.58	3.42	**4.53**

The cached curve flattens while the fresh one keeps falling. At 400 permutations, cached RMSE is 4.53 times the fresh-draw RMSE. Past about 20 permutations, more permutations do not reduce rollout noise. More rollouts per coalition are needed instead (§5).

Semi-synthetic scaling to larger band counts. Exact Shapley with twelve bands already requires 4,096 coalition evaluations. We therefore generate games matched to the real data in player-value scale, interaction ratio, and fraction of negative values. Closed-form Shapley values match exhaustive enumeration in the six-player check.

The scaling depends on how interaction changes with player count. With constant per-player interaction variance, 400 permutations give RMSE 0.024–0.032 over four to twenty players. Reaching RMSE 0.01 needs about 3,200 permutations with at most seven players and 6,400 beyond that. If interaction weakens as the pool grows, 400 permutations give RMSE 0.007–0.014, and 400–800 permutations suffice. The constant-interaction case is the conservative one. **At larger band counts,**

where permutation sampling is required, our 400-permutation setting would be roughly $3\times$ less accurate than Table 5 suggests. Matching the smaller error requires $16\times$ more permutations, not a different estimator. These are calibrated synthetic games, not measured environments. Permutation error depends on game scale and interaction structure, and we did not measure these particular games.

A.8 WITHIN-RUN SHADOW GROWTH

§4 varies the size of a 278-memory pool assembled from twelve sources. This measures how much of a pool of size N a retrieval budget k can reach. It does not isolate accumulation within one agent because larger prefixes also contain more task types. A memory written for a task type none of the 31 audit goals covers will go unretrieved whether or not the pool is growing, so the mixed-source sweep conflates two mechanisms: an agent's later memories crowding out its earlier ones, and a pool acquiring memories the audit workload never asks about.

Two of the twelve sources are single-agent reflexion runs long enough to sweep on their own (66 and 60 memories), and within a source the order is write order. Holding the goals, the retriever and k fixed, and taking prefixes of one run, no new task type ever enters:

sweep	k	smallest	largest	rise	mechanism isolated
`reflexion_dag` (66)	3	0.38	0.79	+0.41	accumulation only
`reflexion2` (60)	3	0.25	0.78	+0.53	accumulation only
mixed-source prefix	3	0.38	0.82	+0.44	accumulation + heterogeneity
`reflexion_dag` (66)	8	0.00	0.42	+0.42	accumulation only
`reflexion2` (60)	8	0.00	0.43	+0.43	accumulation only
mixed-source prefix	8	0.00	0.52	+0.52	accumulation + heterogeneity

The shadow rises in every single-source sweep, and at $N{=}66, k{=}3$ a single agent's own accumulation reaches 79% against the mixed prefix's 82%. Heterogeneity therefore contributes roughly 3 points of a 41-point rise, or about a tenth at $k{=}8$, and the growth this section names is accumulation. Two limits on the claim. These are runs of 60–66 memories, an order of magnitude below the 278 where we report 87–92%, so the single-source evidence is for the onset of the effect rather than its asymptote. And write order within a source is the order the memories were appended, which we take as the order they were written; we did not record timestamps, so a source that was itself assembled out of order would not be detectable here.

A.9 STANCE GATE ON THE LOGGED DAG

Appendix B.3 validates the stance gate on constructed parent sets, where the intended relation of each parent is known. The constructed edges test whether the gate recovers known parent relations; they do not test its behaviour on naturally written memory. We ran it over every edge of the branched ALFWorld DAG: 20 agent-written memories, 28 logged edges, 16 children with parents, and 4 logged sources. We extract each child's claim once with the paper's model and then judging every (claim, parent) pair.

entailment judge	edges kept	sup./con./unrel.	fallback	bands
Claude Sonnet-4.5 (the paper's)	5/28 (18%)	5 / 4 / 19	12/16	16
Claude Haiku-4.5	7/28 (25%)	7 / 9 / 12	10/16	14

We report three measurements. *(i)* The stance-gated partition differs substantially from the source-rooted partition used in the main experiments. With 12 of 16 children having no SUPPORTing parent, the fallback rule promotes each to a source, so the band count rises from 4 to 16, mostly singletons, and agrees with the source-rooted partition on only 6 of 20 memories. At 16 singleton players band attribution is item attribution under another name, which removes the reason for coarsening. We therefore do not use the stance-gated partition for the reported results. *(ii)* Across the logged DAG, 82% of exposure edges fail the stance check, consistent with the distinction between exposure and causal use. *(iii)* The two entailment judges agree on the three-way relation for 20 of 28 edges

(71%), and on the binary SUPPORT/non-SUPPORT decision the gate uses for 26 of 28 (93%); the disagreement is mostly CONTRADICT versus UNRELATED, which the gate discards.

One caveat on all three. A claim extracted from a child *alone* and then compared against a parent is a demanding test for naturally written lessons that share a task but not a proposition, so a high UNRELATED rate is partly a property of our prompt rather than of the memory. We do not know how much, and disentangling it would require human labels that we do not have. This is the same limitation Appendix E.1 records for the edge weight generally.

A.10 SENSITIVITY OF THE EDGE WEIGHT TO THE MODELS IT IS BUILT FROM

The stance-gated rule has two model-dependent parts. We originally used Claude Sonnet-4.5 for entailment and `amazon.titan-embed-text-v2` for similarity. We reran each part with open substitutes while holding the other fixed.

Degree term (embedder). Reusing the recorded entailment relations and recomputing only $s(\cdot,\cdot)$, the intended check holds in $3/3$ scenarios for all three embedders. What does not transfer is *magnitude*: Titan gives weights of $0.81/0.19$ on the cooling scenario where `bge-small-en-v1.5` gives $0.57/0.43$ and `all-MiniLM-L6-v2` gives $0.66/0.34$. The ordering is embedder-invariant here, but the spread is not. We therefore interpret a within-band provenance split as a ranking rather than as a proportion.

embedder	cooling	heating	cleaning
`titan-embed-text-v2` (API, original)	$0.81/0.19$	$0.72/0.28$	$0.71/0.29$
`BAAI/bge-small-en-v1.5` (open, CLS)	$0.57/0.43$	$0.54/0.46$	$0.55/0.45$
`all-MiniLM-L6-v2` (open, mean)	$0.66/0.34$	$0.64/0.36$	$0.68/0.32$

Gate term (entailment judge). Holding claim extraction fixed and varying only the entailment call, Claude Sonnet-4.5, Claude Haiku-4.5 and `Qwen2.5-Coder-7B-Instruct` run locally in `bf16` return *identical* relations on all 12 constructed edges, so the gate is not an artefact of one model or one model family on unambiguous pairs. On the real DAG the same three-way agreement falls to 71% and the gate-relevant binary to 93% (Appendix A.9); agreement is high where the relation is clear and lower where it is not.

Reproducibility. With an open-weight embedder and an open-weight judge, the stance-gated rule is computable without API access; this makes the band construction reproducible, though it does not remove the rollout cost of the attribution itself.

A.11 THE BANDS WE ATTRIBUTE ARE THE CANONICAL PARTITION

The band construction in Appendix A.5 takes the sources of the generation DAG, not an arbitrary antichain. We verify on the logged pools that this recovers the partition our results actually use. On the branched ALFWorld DAG (20 memories, 28 edges) there are exactly 4 sources, one per recorded family, and **every one of the** 20 **memories has all of its source-ancestors inside its own family**, so the family labels the attribution runs on *are* the canonical source-rooted bands, with no residual freedom. The same DAG also shows why the stance-gated variant needs multi-parent handling: 12 **of the** 20 **memories have more than one parent**, so overlapping descendant sets are the common case.

Partition exception. Three of our pools (`dc`, `dcfine`, `bestk`) were partitioned by *lineage level* instead of by source, a departure from the construction in Appendix A.5; they are the pools whose band labels read `L`i`_`j, not `fam`i. We keep them because they are the only place we have the same pool at two granularities, which makes the granularity comparison of Appendix A.3 controlled; they are not instances of the canonical rule, and are excluded from claims about the canonical source-rooted partition. We also do not claim the canonical partition is the more stable one: across the eight band pools the measured-best band survives a change of split on 5, and the split by partitioning scheme is mixed (2 of 3 level-based, 3 of 5 source-based), which is too small a sample to separate the schemes.

A.12 Measured cost and approximation behavior of band-level Shapley

Our evaluated pools contain 8–16 memories and 4–7 Shapley players after grouping memories into bands. The grouping therefore reduces the number of units evaluated in each sampled permutation. It also changes the estimand: a band value is not a cheaper estimate of the corresponding item values.

We check the approximation directly in Appendix A.7. On two seven-player pools with complete coalition tables, RMSE against exact Shapley is 0.006 on one pool at 400 permutations and within 0.006 of that value on the other. In a cached-noise simulation, the error flattens after about 20 permutations, so additional rollouts per coalition are more useful beyond that point. The realised episode counts for every experimental leg are reported in Appendix D.14.

Grouping does not make effects larger. Across all evaluated attributions, mean $|\phi|$ is 0.059 at band level and 0.072 per memory. Within matched pools, band effects are larger in two cases and smaller in two cases. We therefore use bands for coverage and provenance, not because they produce larger effects.

A.13 Measured retrieval reach

Setup and result. Continual-learning work asks how a growing history changes performance (Li et al., 2026b). We instead measure how much of a growing memory pool an attribution audit can see. We use 278 self-written memories from 12 interleaved sources and $P{=}31$ held-out goals. At $N{=}278, k{=}3$, the shadow is 87–92% across token overlap, two dense encoders (Reimers & Gurevych, 2019), and MMR (Carbonell & Goldstein, 1998). At $k{=}16$, it is 54–62%. A power curve fits the measured $k \times N$ grid with RMSE 0.039–0.052 and $R^2 \geq 0.93$. A logarithmic curve fits as well on three of four retrievers (Appendix D.5). We therefore use the curves to summarize the measured range, not to predict larger pools.

The measurement has three qualifications. First, within-run prefixes still show growth, so source heterogeneity does not explain the full pattern (Appendix A.8). Second, a logarithmic model fits as well as the power curve on three retrievers, so the exact functional form is not identified. Third, if the number of audit tasks grows with the pool, the shadow rises and then plateaus (Appendix D.6).

The measured curves exceed the counting bound. At $N{=}278, k{=}3$, the bound accounts for a 66.5% shadow, compared with the measured 92%. Appendix D.2 attributes the remaining gap jointly to retrieval concentration and a narrow workload. Increasing k can also change task performance: on ALFWorld, $k{=}1 \to 10$ changes lift from $+2.8$ to -8.3pp, while ScienceWorld improves. Appendices C.3 and C.5 give the protocol and retrieval-frequency measurements.

Held-out checks and parameter uncertainty. We test the fitted curve by holding out larger pool sizes and by holding out retrievers.

Held-out prefix. Refitting on $N \in \{32, 64, 100\}$ only and predicting the four larger pools $N \in \{150, 200, 250, 278\}$ gives worst-case absolute error 0.050 (token overlap), 0.111 (MiniLM), 0.073 (bge-small) and 0.091 (MMR), with means 0.023–0.057. The fitted exponent moves from $+0.475$ to $+0.472$ on token overlap and by at most 0.14 on any retriever.

Held-out source. Fitting on three retrievers pooled and predicting the fourth gives worst-case errors of 0.118–0.199, appreciably worse than held-out N. The curve shape transfers better than its constants. A practitioner should therefore refit the curve for a new retriever.

Parameter CIs (nonparametric bootstrap over grid points, 2000 resamples):

retriever	c	a	b
token overlap	$0.71\ [0.47, 0.91]$	$+0.83\ [+0.75, +0.93]$	$+0.475\ [+0.41, +0.56]$
MiniLM	$1.49\ [1.04, 2.28]$	$+0.57\ [+0.47, +0.67]$	$+0.493\ [+0.39, +0.58]$
bge-small	$1.90\ [1.22, 2.65]$	$+0.65\ [+0.56, +0.74]$	$+0.379\ [+0.30, +0.46]$
MMR	$2.01\ [1.27, 2.94]$	$+0.60\ [+0.49, +0.70]$	$+0.417\ [+0.32, +0.51]$

Within this power-curve parameterization, every bootstrap interval has $b < 1$. The intervals overlap substantially across retrievers, and the logarithmic alternative fits similarly. We therefore do not interpret an individual exponent as a general result.

The descriptive fit. We fit a power curve separately for each retriever by ordinary least squares in log space. The fit uses retrieval budgets 3, 5, 8, and 16 and pool sizes from 32 to 278. We drop smaller pools because reach is capped by pool size there, which would bias the pool-size exponent downward. Table 8 reports the scale c, budget exponent a, and pool-size exponent b.

Table 8: Power-curve fit over the measured grid of four retrieval budgets and seven pool sizes.

retriever	c	a	b	R^2
token overlap	0.71	0.83	0.475	0.968
MiniLM (dense)	1.49	0.57	0.493	0.936
bge-small (dense)	1.90	0.65	0.379	0.948
MMR (diversity-aware)	2.01	0.60	0.417	0.930

Scope. These constants describe this measured grid. We do not extrapolate them beyond it. MMR applies a redundancy penalty and shows the same measured trend, but this does not cover retrievers with an explicit coverage constraint.

B THE METHOD IN DETAIL

B.1 MEMORY-ATTRIBUTION PROCEDURE

Contribution (2) is a methodology, so we state it procedurally as well as in prose. Nothing here is new relative to §5; it is the same content in the form a reimplementation needs.

Algorithm 1 Memory attribution given a memory pool, disjoint derivation and evaluation task sets, a retriever, and the generation DAG.

1: **Partition the pool by what retrieval surfaces.** Run the retriever over the derivation tasks. Memories retrieved at least once form the *visible* set; the rest form the observed shadow of Def. 3.2. Item-level retrieval-only values are reported on the visible set. Lineage bands provide a group-level unit for the rest of the measured pool.
2: **Visible channel (MEMATTR-V).** For each evaluation task, enumerate every subset of the visible set, estimate each subset's value with the chosen number of rollouts, and return the Shapley value per memory. The subset enumeration is exact, while each value is estimated from finite rollouts.
3: **Build bands.** Take the memories written with no retrieved context as sources, as specified in Appendix A.5 (three pools depart from this and are marked in Appendix A.11). Assign each memory to the source with the largest cumulative lineage weight. The reported results use the ancestry of the logged DAG: every memory in our pools has one source ancestor, so the largest-weight choice never binds, and the stance gate is validated but not deployed, since on these DAGs it keeps under a fifth of the edges (Appendices A.9, A.11). A pool with genuinely multi-source memories would exercise the weighting, and we have not measured that case.
4: **Band channel (MEMATTR-B).** Run permutation Monte-Carlo over the bands with a preselected number of permutations, cache coalition values, and return one value per band. Choose the permutation and rollout budgets before running; Appendix A.7 reports their measured effect on approximation error.
5: **Measure evaluation targets.** For each memory and each band, insert it alone into an empty pool and measure success; leave-one-in, on the *evaluation* tasks only.
6: **Score, and report the unit.** Rank-correlate each estimator against its evaluation target. Report per-memory values only conditional on the stated task distribution, and per-band values for the shadow; never a per-memory number for a never-retrieved memory.
7: **Calibrate the instrument.** Before claiming any memory-editing effect, run a null arm whose edit set is empty; if its measured effect exceeds the effect of interest, the environment cannot resolve the claim (Appendix C.6).

Step 1 distinguishes memories with item-level retrieval support from the rest of the pool. Step 7 provides a null calibration for downstream editing experiments.

B.2 COMPOUNDING CREDIT PROPAGATION

A self-improving agent's memories are not independent items: it reflects on episodes that were themselves run with earlier memories in context, so memory m_3 can shape m_2, which shapes m_1. Credit therefore *compounds along the lineage*: the memory responsible for a failure may be an ancestor that never appeared in the failing trajectory, the case no trajectory- or retrieval-level method can express. Propagating credit backwards along the DAG makes such an ancestor attributable. The optional stance-gated rule can avoid charging a same-topic parent that argued the *opposite*. Def. 3.1 gives the target: remove the memory and rerun generation for every descendant. This attributes a descendant's measured value to the ancestors on which its existence depended. It requires a generation rerun; a fixed-pool ablation does not recompute descendants.

The discrete operator, for when the re-run is unavailable. Re-running generation needs the write process to be replayable. When it is not, the indirect component can be estimated on the logged DAG instead. This is the estimator behind Table 6. Process memories in reverse topological order. Start each memory's accumulated credit at its measured leave-one-out effect on the failing task. Add an equal share of each child's accumulated credit to every parent. The indirect score is the accumulated credit beyond the direct effect, discounted by 0.8; the total score adds the direct and indirect components.

The discount is applied once after propagation rather than at every edge. Applying it at every edge repeatedly discounts deep roots and can make a nearer descendant outrank the source. The once-only discount avoids this depth penalty. The operator also splits credit uniformly across parents. The optional rule instead weights parents by stance. Uniform splitting is the fallback when no stance signal is available. Appendix B.3 shows its limitation: a same-topic parent that contradicts the child receives an equal share.

Example on a logged lineage. On an ALFWorld failure (*"cool some bread and put it in countertop"*) retrieval surfaces one relevant memory, whose advice is wrong, so every retrieval-only estimator charges that child and stops. Its parents were not retrieved, and one is the ancestor whose stance the child inherited. A same-topic *sibling* saying the opposite is correct and must not be charged, but a similarity-based weight charges it most. In contrast, the stance gate gives it zero weight (Appendix B.3).

The evaluation target is the leave-one-in utility of §3, applied to the estimator's unit. Nothing is hand-annotated; every value is a measured change in agent success. An annotator asked which self-written memory mattered would have to guess the quantity being evaluated.

B.3 MULTI-PARENT LINEAGE: VALIDATING THE EDGE WEIGHT

A concrete case from the ALFWorld pool, with the memory texts verbatim. The failing episode (*"cool some bread and put it in countertop"*) retrieves exactly one relevant memory, the child: *"Allow the bread to cool to room temperature on the countertop before moving it, the fridge is unnecessary."* That advice is wrong (nothing ever cools) and it is all that entered the prompt, so every retrieval-only estimator charges the child and stops. The lineage has parents, none retrieved here, one of which reads *"When cooling, you do NOT need the fridge, it cools on the countertop"*: the ancestor whose stance the child inherited. A same-topic sibling, *"To cool an item, take it to the fridge and use the fridge,"* is *correct* and must not be charged. A similarity-based edge weight nevertheless charges it most (cosine 0.36 vs. the responsible parent's 0.32), while the entailment gate gives it 0 for contradicting the child. The attribution names a lineage absent from the failing trajectory. A method that uses only that trajectory cannot name this lineage.

The stance gate is one modelling choice in the optional weighted construction, and it matters because under the setup of §3 every memory has up to k parents, so the multi-parent case is the normal case rather than an edge case. We therefore tested candidate weight functions on controlled parent sets, where the correct answer is known by construction: a child memory states a claim, one candidate parent shares that claim, one is *same-topic with the opposite stance* (a decoy), and the remainder are unrelated. A weight function is correct on such a set if it charges the stance-sharing parent and **spares the decoy**, which is the question of whether blame concentrates on the responsible ancestor or smears onto innocent ones. Table 9 summarises the four candidates we tested.

Table 9: Four candidate edge weights on controlled parent sets. The columns give the candidate, its status, and the observed behaviour.

candidate weight		**outcome**
content-overlap LLM judge	fails	requires the child's claim inside the question asked of the parent, so it only works when the answer is already known; circular, not an estimator
generation-time ablation ($K{=}5$)	fails	drop a parent, re-generate the child, measure the change; re-generation variance charged the *unrelated* parent at 0.83
embedding cosine alignment	fails	tracks topic, not stance: the same-topic *opposite-stance* decoy scored 0.36 against the responsible parent's 0.32
symmetric entailment (Appendix A.5)	works	gates on whether the parent supports or contradicts the child's own claim, asked as the same neutral question of every parent

Isolation. On a cooling scenario, cosine ranks the decoy first (0.36 vs. 0.32, then 0.23 and 0.09), i.e. it charges the parent whose lesson was *correct*. The entailment gate assigns SUPPORT$\rightarrow$ 1.00 to the responsible parent and 0.00 to the decoy and both unrelated parents, because the decoy *contradicts* the child. This holds on 4 of 4 topics tested (cooling, heating, cleaning, light-examine).

The continuous split. The gate alone gives $0/1$ weights, which cannot divide credit between two parents that both pointed the same way. The similarity term does: on a cleaning scenario with two supporting parents at similarity 0.88 and 0.36, the weights are 0.71 and 0.29, while a contradicting parent at similarity 0.76, which a similarity-only weight would charge heavily, receives 0.00. Robustness $3/3$ on the scenarios with two supporting parents.

Status and limits. These are constructed parent sets, not a sample of naturally-written children: $4/4$ isolation and $3/3$ continuous-split is a unit test of the weight function, not an attribution result, and we present the stance-gated rule accordingly: as a stated convention with a controlled validation. Three specific gaps remain. The gate is an LLM call (symmetric across parents and never shown the child's outcome, but not LLM-free), so it needs a consistency check at scale. The degree term used a proprietary embedder in this test and should be re-run with an open one for reproducibility. And the quantity it measures is *stance inheritance*: a child inherits what its parent argued. For compounding credit along a lineage the two coincide when a child's error is its parent's stance restated, which is the case these scenarios were built to isolate, and we do not claim they coincide in general.

C BENCHMARK, PROTOCOL, AND INSTRUMENT

C.1 BENCHMARK ARTEFACTS AND TABLE-READING NOTES

The released artefacts are, per environment: the **agent-written memory pool**; the **generation DAG** recording which memories were in context when each memory was written, the prerequisite for band-level attribution, which existing agent frameworks generally do not log; the **measured evaluation targets**, both per-memory leave-one-in and per-band leave-one-band-in; the **retrieval configuration** (retriever, budget k, and the source-interleaving used to build the 278-memory pool, Table 12), and the **split protocol**, under which an attribution is derived on one task split and evaluated on a disjoint one. Empty-pool success $v(\emptyset)$ ranges 0.190–0.500 across pools.

MEMATTR-V tracks measured leave-one-in utility wherever the environment has measurement room, reaching ρ up to 1.00 on ScienceWorld and 0.90 on τ-retail. Separately, and not as a fourth environment, the semi-synthetic SWE probe reaches 0.95; we keep that number out of every aggregate in this paper because its memory is gold-patch-guided rather than agent-written, so it measures whether the visible channel transfers to a code domain and not whether it recovers the value of memory an agent wrote. MEMATTR-B beats the M4 granularity baseline on 8 of 12 pool/splits, and perm-MC goes negative on 2 where M4 goes negative on 8, so M4 turns *negative* on four pools, i.e. ranks the shadow backwards. M2 (the standard flat LOO) *reverses* the evaluation target on ALFWorld, which motivates an interaction-aware estimator. Three per-environment caveats behind

the marked cells of Table 11 (*, †, and ScienceWorld's weakest MEMATTR-V split) are stated in Appendix D.10.1.

C.2 WHY THE STATISTICAL UNIT IS THE TASK

These agents run at temperature 0 and are close to deterministic: on ALFWorld, 78 of 90 (arm, task) groups were identical across all eight seeds. The remaining 12 groups differ across seeds. The structural counterfactual M^{-m} of Def. 3.1 is not quite unique on those cells, so the "same (τ_j, u_j)" reading is an approximation there rather than an identity. If two interventions produce the same retrieved prompts, their underlying outcome distributions are still the same. A finite cached rollout can show a small nonzero difference because the two empirical means use different random draws. Every ρ and p in the paper therefore sits on a noise floor above the deterministic ideal. Appendix C.6 measures the same floor at $+0.054$ from the other direction.

Seeds are near-duplicates, not independent samples. A per-cell test therefore treats one observation as eight. For one pruning policy, the per-cell test gives $p{=}0.039$, while the task-level test gives $p{=}0.45$. The per-cell test would therefore produce a false positive. We compute every p-value over tasks. More tasks, not more seeds, increase the effective sample.

C.3 SHADOW MEASUREMENT

We measure the shadow on 278 memories the agents wrote themselves, from 12 sources interleaved so the source mix is constant at every prefix, against $P{=}31$ held-out goals. It needs no rollouts, since the shadow is a property of retrieval. To test whether the shadow is caused by weak retrieval, we repeat the measurement with four retrievers, including MMR, which explicitly penalises near-duplicates.

C.4 RETRIEVAL INVARIANCE UNDER MEMORY DELETION

We test whether a memory that baseline retrieval never surfaces can still change the retrieved prompt when it is deleted. The test uses the same 278-memory pool and 31 held-out goals as the shadow measurement. For each retrieval budget, we first record the ordered top-k memory IDs returned by the deployed deterministic token-overlap retriever. We then remove each never-retrieved memory separately, rerun all goals, and compare the ordered lists with the baseline.

k	**baseline visible**	**shadow tested**	**changed shadow comparisons**	**visible controls changing a list**
3	23	255	0/7,905	23/23
16	107	171	0/5,301	107/107

No single deletion from the measured shadow changes any retrieved list. As a positive control, deleting every memory that appears in a baseline list changes at least one query's list. The check therefore detects retrieval changes when they occur. It establishes invariance for these single-memory deletions and this retriever; it does not cover arbitrary multi-memory interventions or other retrieval implementations.

C.5 RETRIEVAL RARITY AND COVERAGE BUDGET

The retrieval logs contain little support for most memories. Among memories retrieved at least once, 33% appear on only one or two evaluation tasks across six pools, and the median retrieval count is 3. Appendix D.10.2 gives the per-pool counts.

Increasing the retrieval budget exposes more memories. Table 15 (Appendix D.10) measures the tradeoff: on ALFWorld $k{=}1 \to 10$ cuts the shadow $94\% \to 61\%$ but moves pool lift $+2.8 \to -8.3$pp (context dilution) whereas on ScienceWorld lift improves.

C.6 NULL-ARM CALIBRATION, IN TWO ENVIRONMENTS

A flat result may reflect either method failure or insufficient measurement resolution. The check that distinguishes the two is an *empty*-prune arm: the same memory set as keep-everything, re-executed under a separate cache key, so its true effect is 0 and whatever it measures is the floor. We ran it in both environments where we prune, and the two floors differ enough that borrowing one for the other would have misled us.

environment	measured floor	cells flipping	consequence
ALFWorld	$+0.054$ (p=0.06)	—	above *every* pruning effect there
τ-retail	$+0.000$ / $+0.013$	18% / 16%	below both P3 (-0.075) and P5 ($+0.062$)

On ALFWorld the floor exceeds every pruning effect we measured, so those results cannot distinguish method effects from environment-level measurement noise; deleting *all* 16 memories in that pool does not lower success, so there is nothing there for pruning to improve. On τ-retail the floor is about a quarter as large and sits below both arms of the confirmatory test, so the reversal reported in §6.3 is not a floor artefact: P5's $+0.062$ is well above it, so its advantage over P3 is not explained away by run-to-run variation. A floor borrowed from ALFWorld would have suggested otherwise, which is the reason to measure one per environment.

The τ-retail arm also separates two quantities a single number hides. Its *rate* difference is ≤ 0.013, but 16–18% of individual cells flip between two identical configurations (b=7, c=7 and b=6, c=7 of 80). Per-cell nondeterminism is therefore substantial and cancels only in aggregate, which is why every test here is a paired McNemar on discordant cells and why the per-cell analysis of §6.2 would be overconfident by roughly the factor Appendix C.2 quantifies. We recommend reporting such a calibration alongside any memory-editing result, in the environment the result is claimed in; without it a flat row and an unresolvable measurement look identical.

D ADDITIONAL RESULTS

D.1 BAND GAMES WITH NON-POSITIVE TOTAL VALUE

Half of our band games have a non-positive total. In these cases the Shapley vector is a pure interaction decomposition summing to a non-positive value, and correlating it against a *solo* quantity is not obviously meaningful. Worse, in the two Llama pools $v(\text{all}) = v(\emptyset) = 0$ and every band value lands in the third decimal place, so the ranking is decided by perturbations at the noise floor. These two cells also carry two of the highest band ρ values. We therefore split the cells by whether their game is resolvable at all.

cell group	cells	mean ρ	positive
all	16	$+0.448$	$12/16$
game total > 0 (a real effect to attribute)	8	$\mathbf{+0.370}$	$\mathbf{5/8}$
game total ≤ 0 (nothing net to attribute)	8	$+0.527$	$7/8$

Performance by game total. On the eight cells where memory has a net effect, the exact band channel averages $+0.370$ with five of eight positive; on the eight where it does not, it averages $+0.527$ with seven of eight positive. The four cells whose largest absolute player value is below 0.03 average $+0.550$. The higher score on smaller effects is consistent with rank instability near the noise floor. We therefore exclude these cells from the aggregate.

Aggregation rule. We report band aggregates on the $\text{total} > 0$ subset, where the mean is $+0.370$ and $5/8$ cells are positive. We retain all cells in the per-cell table and mark the non-positive games. For the two Llama pools, $\rho = +0.83$ and $+0.82$ are computed from Shapley values whose largest magnitude is 0.02; these values mainly reflect tie-breaking near the noise floor.

D.2 Effect of workload diversity on retrieval reach

The reach measurement can reflect both retriever concentration and the held-out workload. Our 31 ALFWorld goals are homogeneous: they are pick-and-place variants spanning only 6 verb-phrase types, with some literal duplicates. A narrow workload could therefore reduce reach even if the retriever behaved as intended.

The test that separates the two holds the goal *count* fixed and varies the *diversity*, since adding goals changes both at once.

	goals from 1 type	**goals from n types**	**ratio**
$n = 4$ goals	$R = 7.9$	$R = 10.3$	$1.31\times$
$n = 6$ goals	$R = 10.1$	$R = 14.1$	$1.40\times$

Retriever concentration and workload diversity. Spreading the same number of goals across types raises reach by 31–40%, so part of the narrow reach is that the workload asks few distinct things. On the full set, reach is $R = 23$ against a budget cap of $kP = 93$ and a pool of $N = 278$: the retriever uses a quarter of the slots available to it, and a $1.4\times$ diversity effect does not close a $4\times$ gap. Reach per goal falls from 1.00 at one type to 0.67–0.74 at six. Each new goal therefore surfaces fewer previously unseen memories.

Scope of the reach measurement. The measured curve depends on both the retriever and the workload. Most of the pool is unseen because retrieval concentrates on a few memories and the goals cover few task types. A more heterogeneous workload is needed to separate these effects further.

D.3 Size-matched random-partition control

MemAttr-B does two things at once: it *coarsens* the pool into a few groups, and it coarsens along the *generation lineage* using the construction in Appendix A.5. Coarsening alone reduces the number of players and creates a group-level unit that can include a never-retrieved memory. Neither property requires the lineage. This control tests whether lineage structure improves attribution accuracy beyond the effect of coarsening.

Design. On the ALFWorld `branched` pool (20 memories, canonical partition $[5, 5, 5, 5]$) we build three *random* partitions with the same group count and the same group sizes, ignoring $\mathcal{G}$ entirely, and run the identical exhaustive band game against the identical measured leave-one-band-in target. Same pool, same evaluation games, same k, and the same `seeds`$=3$. An initial `seeds`$=1$ run was not comparable to the canonical number.

partition	**ρ vs. measured target**
canonical (source-rooted lineage)	$+0.600$
random, sizes matched, run 0	$+0.775$
random, sizes matched, run 1	$+0.800$
random, sizes matched, run 2	$+0.316$
random mean	$+0.630$

The lineage does not show an accuracy advantage. The canonical value sits *inside* the random range $[+0.316, +0.800]$, two of three random partitions exceed it, and the random mean is slightly higher. With three draws this does not show that random partitions are better; it shows that this experiment provides no evidence that lineage-based grouping improves attribution accuracy over size-matched random grouping. On this pool, what MemAttr-B recovers is attributable to coarsening into four groups rather than to coarsening *along the lineage*.

Rationale for the lineage partition. Three properties are independent of this accuracy result. *(i)* Each sampled permutation evaluates one marginal contribution per player, so player count matters regardless of how the groups are formed. *(ii)* A band has a group-level counterfactual even when

one of its members is never retrieved. Appendix D.4 shows the same boundary from the other side: a retrieval-defined label is undefined on ten of sixteen memories. *(iii)* The source-rooted partition comes directly from the logged generation DAG and adds no hyperparameters. A random partition needs a seed, while topic clustering needs a similarity threshold and a cluster count.

These properties support reproducibility and provenance, not an accuracy advantage. A further difference matters for the generation channel. A random group can be ablated but not regenerated because it has no lineage root. The generation-side counterfactual in Def. 3.1, and the measurement in Appendix D.7, are therefore available only for lineage bands. This is why logging lineage remains useful even though the random-partition control shows no accuracy advantage.

Scope. One pool, one split, three random draws, at $n{=}4$ players where Appendix D.8 shows no single ρ can be significant. A larger evaluation, with more pools, more random draws, and topic- or level-based partitions alongside random ones, would be needed to distinguish these constructions.

D.4 Xiong et al.'s quality label as a baseline

Xiong et al. (2026) are the closest prior work and their central device is cheaper than ours: use *future task evaluations* as free quality labels for stored records. For each memory, we average the full-pool outcomes over evaluation tasks that retrieved it. No new rollouts are needed because the deterministic retriever lets us reconstruct support from the previously measured full-pool outcomes. The reconstructed support counts match the recorded counts on $6/6$ memories in both splits. An earlier attempt used the dataset's task description instead of the environment goal and reproduced only $1/6$, so we discarded it.

	scored on	**undefined on**	ρ **Xiong**	ρ **MemAttr-V**
ALFWorld split a	5 of 16	10 never retrieved	$+0.667$	$\mathbf{+0.975}$
ALFWorld split b	5 of 16	10 never retrieved	$+0.667$	$\mathbf{+0.900}$

Result. The label reaches $\rho = +0.667$ on both splits and provides a low-cost ordering of retrieved memories. It is undefined for the ten of sixteen memories that no task retrieved because its averaging set is empty. The difference from MemAttr-V on the five retrieved memories is not significant (Appendix D.8). The main distinction is coverage: lineage-band attribution defines a unit for the part of the pool that this retrieval-defined label cannot score.

D.5 Power-law model comparison

We compare the descriptive power curve in Appendix A.13 with logarithmic, saturating, and no-concentration alternatives on the same grid of budgets, pool sizes, and measured reach. We fit each in log space and compare adjusted R^2 and AIC.

adjusted R^2 (AIC gap to best)	token	MiniLM	bge	MMR
power curve	$\mathbf{0.964}$ (0)	0.928 ($+3.5$)	0.941 ($+0.8$)	0.922 ($+3.3$)
logarithmic pool-size curve	0.961 ($+2.0$)	$\mathbf{0.936}$ (0)	$\mathbf{0.943}$ (0)	$\mathbf{0.930}$ (0)
saturating curve	0.218 ($+86$)	0.425 ($+62$)	0.222 ($+73$)	0.312 ($+64$)
no-concentration baseline	-0.108 ($+94$)	-0.069 ($+77$)	-0.576 ($+91$)	-0.413 ($+83$)

The power and logarithmic curves beat the no-concentration baseline and the saturating form by 60–94 AIC. The logarithmic curve matches or beats the power curve on three of four retrievers, with AIC gaps of 0.8–3.5 that this grid cannot resolve. Thus, the measured data show strong retrieval concentration but cannot tell which curve is correct.

D.6 Shadow growth with the audit budget coupled to the pool

The measurement of §4 holds the audit set at 31 tasks while the pool grows, so the audit budget per memory falls. In deployment, the task count and pool size can grow together. We reran the same retrieval-only measurement while scaling the audit set from one task at pool size 8 to 31 tasks at

pool size 278. This keeps the counting bound roughly fixed at 0.33 for retrieval budget 3 and 0.89 for budget 8. No rollouts are required.

k	N	P	token	MiniLM	bge	MMR
3	8	1	62%	62%	62%	62%
3	32	4	84%	88%	84%	84%
3	100	11	93%	86%	90%	87%
3	278	31	92%	87%	88%	88%
8	8	1	0%	0%	0%	0%
8	32	4	69%	69%	69%	66%
8	100	11	81%	72%	71%	68%
8	278	31	78%	71%	77%	73%

The coupled experiment shows two effects. First, every retriever moves from a small shadow at N=8 to a large one at N=278 even when the audit ratio is fixed. Second, the fraction is already near its measured ceiling by $N \approx 100$ and then flattens. A constant audit ratio therefore leaves a large share of the measured pools unexamined. We make no claim about the limiting fraction.

D.7 Direct regeneration check

Most experiments in this paper keep the written memory pool fixed. This experiment repeats the write process. It removes each lineage root, regenerates its descendants with fixed tasks and seeds, and then evaluates the new pool. The resulting performance change measures the generation influence in Def. 3.1 on one pool.

Protocol. On the ALFWorld `offset200` pool (20 memories, 4 canonical source bands) we withhold each band's root and *regenerate* that band's descendants from their original build tasks with the root absent from context, then evaluate the regenerated pool. Regeneration tasks are the build games (`train` split, `pick_clean`[200:220]) and evaluation is the held-out window (`eval_out_of_distribution`, 8 games × 3 seeds); the two are different environment splits, so they cannot overlap. $\mu(M) = 0.417$.

band	$\mu(M^{-\text{band}})$	success change	band Shapley
`fam0`	0.125	$+\mathbf{0.292}$	$+0.063$
`fam1`	0.500	-0.083	$+0.035$
`fam2`	0.333	$+0.083$	$+0.035$
`fam3`	0.250	$+0.167$	-0.007

Largest change. Removing `fam0` and rewriting its descendants changes held-out success by 0.292. This is more than twice the full band game's total value of 0.125 on this pool.

Comparison with band Shapley. Spearman correlation between band Shapley and the regeneration changes is $+0.32$ over four bands, and `fam1` has the wrong sign. At n=4, this correlation cannot be significant (Appendix D.8). MemAttr-B measures the value of an already-written lineage. It does not estimate the result of rewriting that lineage.

Scope. This experiment covers one pool and four bands. Extending it needs one evaluation pass per band plus a baseline pass and one regeneration per descendant, so its cost grows with the pool.

D.8 Uncertainty and aggregate tests

Every ρ in this paper is computed over four to seven players. At this size, a percentile bootstrap is coarse, so we also report an *exact* permutation p by enumerating every relabelling of the estimator scores. The exact test has a hard floor: with four players, the smallest attainable two-sided p is 0.083, so **no four-player cell can be significant at** 0.05 **whatever it scores**, and 17 of our 27 cells have n=4.

Per-cell significance. Across all 27 pool/split cells (both channels), exactly 2 reach exact $p < 0.05$: MemAttr-V on ALFWorld-Ministral a (ρ=+0.97, p=0.033) and on ALFWorld-Qwen b (ρ=+0.94, p=0.017). Eleven of the 27 bootstrap intervals contain zero. Individual-cell significance should therefore not be interpreted as evidence for the overall claim.

Table 10: Spearman ρ against measured leave-one-in utility for full Shapley, Shapley with the singleton term removed, and item LOO. The table includes every pool with a complete coalition table.

pool	full Shapley	singleton term removed	item LOO
ALFWorld-Ministral a	$+0.98$	$+0.98$	$+0.80$
ALFWorld-Ministral b	$+0.90$	$+0.70$	$+0.79$
ALFWorld-Qwen a	$+0.80$	$+0.80$	-0.47
ALFWorld-Qwen b	$+0.94$	$+0.94$	$+0.62$
ScienceWorld melt	$+0.27$	$+0.27$	$+0.54$
τ-airline	$+0.69$	$+0.69$	$+0.65$
τ-retail	$+0.82$	$\mathbf{-0.40}$	-0.95
mean	$\mathbf{+0.77}$	$\mathbf{+0.57}$	$+0.28$

Aggregate test. We evaluate consistency across pool/split cells with a sign test. For MEMATTR-B, we report both all cells and the positive-total subset defined in Appendix D.1.

estimator	cells	positive	negative	median ρ	sign test
MEMATTR-V	11	11	0	$+0.89$	$p = 9.8 \times 10^{-4}$
MEMATTR-B, all cells	16	12	3	$+0.73$	$p = 3.5 \times 10^{-2}$
MEMATTR-B, total > 0	8	5	3	–	$p = 0.727$

The visible estimator is positive in all 11 cells. The all-cell band test gives $p = 0.035$, but this result is driven in part by games with non-positive total value. On the positive-total subset, five of eight cells are positive and the sign test gives $p = 0.727$. We therefore treat the band result as descriptive within-split agreement rather than significant aggregate evidence.

D.9 SHARED-TERM ABLATION

Our evaluation target and Shapley estimator share the singleton-coalition term: both include the value of adding one player to the empty pool. Item LOO instead measures removal from the full pool. With 4–6 players, the shared singleton term contributes 17–25% of the Shapley average and none of the item-LOO baseline. This overlap could explain part of the difference in Table 4.

We therefore remove the singleton term from each Shapley value, renormalise the remaining coalition-size terms, and score the result against the unchanged target. The computation uses the same cached coalition values. Table 10 compares the resulting rankings.

The ablation changes the result unevenly across pools. The ablated estimator still correlates at $+0.57$ against LOO's $+0.28$, and on five of seven pools the ranking is unchanged, because with heavily tied targets the singleton term changes values without changing order. On τ-retail, however, the correlation changes from $+0.82$ to -0.40. We therefore quote the ablated mean whenever the shared term could affect the comparison.

D.10 ADDITIONAL EXPERIMENTAL DETAIL

Experimental setup. We evaluate on three environments in which the memory is *entirely agent-written*: ALFWorld (Shridhar et al., 2021), built on TextWorld (Côté et al., 2018); ScienceWorld (Wang et al., 2022); and the airline and retail domains of τ-bench (Yao et al., 2024). On ALFWorld, we use Ministral-14B, Qwen3-32B (Qwen Team, 2025), and Llama3.1-8B (Llama Team, AI @ Meta, 2024). On τ, we use Claude Sonnet-4.5 (Anthropic, 2025). We run the judge baseline with two judge models and report both (Appendix D.17).

Memory is stored and retrieved through a standard agent memory substrate (Chhikara et al., 2025). Pools contain 8–16 memories and 4–5 Shapley players (Table 3). We also run a software-engineering probe on SWE-bench-Lite (Jimenez et al., 2024). Its memory is guided by gold patches rather than written entirely by the agent, so we exclude it from the main comparison (Appendix D.10).

We selected pools with a **lift gate**: the full pool had to beat an empty prompt. The lift did not hold on replication, so the gate is a screening step, not a guarantee that memory helps. The gate also differs

from a Shapley game total. The game uses 4–5 players on one split, so it can be flat or negative when the full pool helps. This occurs on half of our pool/splits (§5, Appendix D.11).

Every p-value is computed over tasks, not (task, seed) cells. At temperature 0, these agents are near-deterministic. A per-cell test inflates n eightfold and changes one policy's p from 0.45 to 0.039 (Appendix C.2).

Table 11: Per-environment metrics with Qwen3-32B fixed; τ uses Sonnet-4.5. Cells use each method's natural metric and are not comparable across rows or columns. M3 and M5 report localisation metrics. The τ M1 entry uses the whole-pool ranking. §Appendix D.17 compares scoring units. ¶Exhaustive band Shapley on the first split.

method	ALFWorld	ScienceWorld	SWE-bench*	τ-bench
shadow (% never retr.)	62–94%	67–83%	33%	33–50%
M1 judge	P@8 0.39/0.33	cov 92–100%	3/3	ρ=−0.46§
M2 item LOO	reverses	9,6 of 12	lift 1.0 vs 0.17	LOI ±0.2
M3 gen-DAG (shadow cov.)	100%	100%	2/3	58%
M4 granularity (finest level)	1	1 group	4	1
M5 binary search (hits /8)	no	8/8	8/8	n/a
MEMATTR-V visible Shapley (ρ)	0.80/0.94	+0.58 (weak)	0.95	0.90
MEMATTR-B	7/8 sign	+0.82 exact¶	=M3†	+0.50

D.10.1 PER-ENVIRONMENT CAVEATS BEHIND THE MARKED CELLS OF TABLE 11

*SWE is the only environment with semi-synthetic, gold-patch-guided memory. The fully natural version gave floor performance on stably failing tasks, so we report it separately. †On SWE the shadow signal *is* the generation-DAG reach, so MEMATTR-B and M3 coincide.

ScienceWorld has the weakest MEMATTR-V entry in the table, so we mark it instead of bolding it. Its Shapley values sit at the noise floor with n=4 players and ties. The $+0.58$ describes the ranking direction, not accurate per-memory magnitudes. Exhaustive band Shapley reaches $+0.82$ on the first split (Table 5), measured over fewer, larger units. Table 4 reports the perm-MC value, $+0.45$ with range $[0.26, 0.63]$, instead of the exact value. The two numbers therefore do not conflict. This result is consistent with the motivation for using a coarser attribution unit.

D.10.2 RETRIEVAL-FREQUENCY PROFILES

Every attribution run records how many evaluation tasks retrieved each memory. Among memories retrieved at least once, 33% are retrieved only once or twice across the entire evaluation set (six pools, median retrieval count 3). A small number of memories account for most retrievals, while many are retrieved rarely or never.

Table 14 in Appendix D.10 gives the counts: at k=3 the budget would permit kP=93 distinct memories to be seen, and only 23–36 ever are.

D.10.3 MECHANISTIC PROBE PROTOCOL

Zero-shot proxy results. Over five pools in two open-weight families, attention mass and input-gradient saliency straddle zero against the measured target and disagree in sign on most pools. Ten points per signal exclude correlations larger than about 0.2, but not smaller correlations. The position control holds in one model family and not the other.

The whole-pool judge has a different limitation. A whole-pool judge never observes an outcome, so its ranking depends only on the memory text: ours is bit-identical across two disjoint splits in three of four environment/judge pairs, while the measured value changes across splits. Its agreement with the measured target is therefore distribution-dependent, and on one pool it equals MEMATTR-V. This suggests the judge behaves more like a text-based prior than an estimator of task-dependent memory value (Zheng et al., 2023). The result does not establish a general correlation sign (Appendix D.17).

Scope of the proxy result. Training-data attribution answers its cost problem with a surrogate that avoids retraining (Ding et al., 2026; Vitel & Chhabra, 2026); we did not find a zero-shot analogue among the signals we tested. A *trained* surrogate remains possible (Zhang et al., 2026b). Our experiments therefore rule out only the zero-shot signals we tested, not a learned surrogate. Appendix D.14 reports rollout costs.

Per-family numbers and intervals. Averaged over the five pools, attention correlates with measured value at $+0.06$ in Qwen2.5-7B and -0.06 in zephyr-7b. Input-gradient saliency gives -0.03 and -0.08. Pooling both families gives $\bar{\rho}{=}+0.00$ with bootstrap 95% CI $[-0.14, +0.14]$ for attention and $\bar{\rho}{=}-0.05$ with $[-0.21, +0.11]$ for gradient. These data exclude correlations beyond about 0.2, but not smaller correlations. Per-family intervals are wider; Qwen attention spans $[-0.18, +0.31]$.

The *position* control tests whether the protocol detects a known signal. It reaches $+0.64$ $[+0.54, +0.78]$ in Qwen and $+0.11$ $[-0.07, +0.26]$ in zephyr. The zephyr interval contains zero, so that model does not independently validate the probe. Our claim covers averaged attention mass and input-gradient norm on two 7B models over five small pools. It does not cover mechanistic proxies in general.

Attention and gradients require open weights, so the probes run on models that did not write the pools. The pools come from Ministral-14B, Qwen3-32B, Llama3.1-8B, and Sonnet-4.5. The numbers bound what these two signals recover on instrumentable models when reading pools written by other models.

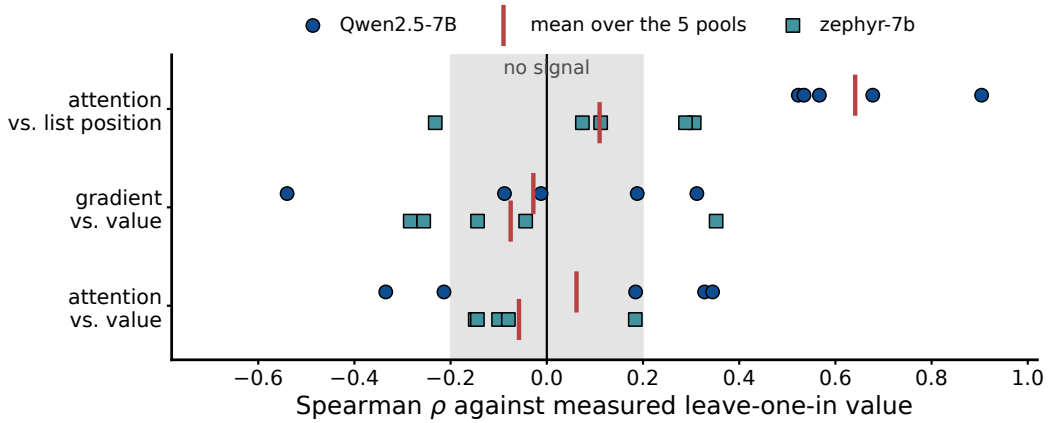

Figure 4: Correlations of attention mass and input-gradient saliency with measured leave-one-in utility. Each point is one pool for one model family; red bars show means. The right panel is the attention-versus-position control.

We use the five pools with exact visible-channel Shapley and present each with 25 held-out query goals. For each query, we build the agent prompt and record every memory's token span. We measure two signals: attention mass from the action-emitting position, averaged over heads and the deepest quarter of layers, and the input-gradient norm summed over the memory span.

Memory order is rotated per query, so each memory visits every list position. This lets us measure the position control in Figure 4. With a fixed order, the first memory draws roughly twice uniform attention, so a correlation with Shapley would partly reflect list position.

Detectable correlation range. Each signal has 5 pools $\times$ 2 model families $=$ 10 points, so we state it as a bound rather than a null. Bootstrap 95% CIs over pools (20,000 resamples):

Table 12: Per-environment retrieval budget, pool-construction tasks, and evaluation splits. All rows use deterministic token-overlap retrieval. Dense and MMR retrievers are used only for the coverage measurement.

environment	k	**pool built on**	**eval split a**	**eval split b**
ALFWorld (`pick_clean`)	3	games 0–15	games 16–22 (7)	games 23–30 (8)
ScienceWorld (melting point)	3	variations 0–11	even $0, 2, \ldots, 10$	odd $1, 3, \ldots, 11$
τ-bench airline	3	tasks 0–11	tasks 12–16	tasks 17–21
τ-bench retail	3	tasks 20–31	tasks 2–11 (derivation)	tasks 32–59 (wide)
confirmatory	3	*as above*	tasks 60–79 (20)	tasks 80–99 (20)
SWE-bench Lite (django, sympy)	3	per-instance*	`django-11797`	—

Table 13: Never-retrieved fraction of the N=278 memory pool over P=31 episodes, by retriever and retrieval budget.

budget	**token overlap**	**MiniLM**	**bge-small**	**MMR** (λ=0.5)
k=3	92%	87%	88%	88%
k=5	86%	79%	84%	81%
k=8	78%	71%	77%	73%
k=16	62%	54%	59%	60%

signal	**Qwen2.5-7B**	**zephyr-7b**	**pooled**
attention vs. value	$+0.06$ $[-0.18, +0.31]$	-0.06 $[-0.14, +0.07]$	$+0.00$ $[-0.14, +0.14]$
gradient vs. value	-0.03 $[-0.30, +0.21]$	-0.08 $[-0.25, +0.15]$	-0.05 $[-0.21, +0.11]$
attention vs. position (control)	$+0.64$ $[+0.54, +0.78]$	$+0.11$ $[-0.07, +0.26]$	$+0.38$ $[+0.18, +0.56]$

The pooled intervals exclude $|\rho| \gtrsim 0.2$ but do not establish zero correlation. Per-family intervals remain wide; for example, Qwen gradients span $[-0.30, +0.31]$. The position control is clear only for Qwen. Its Zephyr interval contains zero, so we do not treat Zephyr as an independent validation of the probe.

*The SWE pool is built per instance, and is the semi-synthetic case noted in Appendix D.10.1; it is scored over 3 seeds on one instance rather than over two task splits, so it contributes a rank correlation but no split-stability number. The ScienceWorld splits stay inside variations 0–11 deliberately: that is the range on which the pool's +33pp lift was measured, and on variations 16–27 every method (ours and the baselines) sits on the floor, which measures the environment rather than the methods. ALFWorld is the only environment where the pool is built by the ReasoningBank-style writer over a contiguous prefix of the task list; elsewhere the writer's task set is listed above explicitly.

The measured shadow far exceeds the counting bound (92% vs. 67% at k=3) because retrieval *concentrates*: it re-selects the same memories rather than spreading over kP distinct ones, so the retrieved count saturates. And the fraction grows monotonically in N for all four retrievers. Table 13 in Appendix D.10 gives the N=278 slice across four budgets; at k=3 the shadow is 92% (token overlap), 87% (MiniLM), 88% (bge-small) and 88% (MMR), and even at k=16 it is 54–62%. Figure 2 shows the growth directly. Two features matter beyond the endpoint values. The curves are ordered by *budget*, not by retriever quality; at a fixed k all four retrievers sit within about five points of each other at N=278, whereas moving k from 3 to 8 shifts every curve by roughly fifteen. And the gap to the counting bound *widens* with N, which is the concentration effect made visible: at k=3, N=278 the budget would permit kP=93 distinct memories to be seen, and only 23–36 ever are.

Workload-specific memory values. A per-memory attribution is defined only for memories retrieval actually surfaces, so its *domain* depends on the task distribution. Computing exact per-memory Shapley on two disjoint task splits of the same pool, the values agree closely where both are defined (all three shared memories match in sign: $(-0.059, -0.056)$, $(+0.072, +0.097)$, $(+0.140, +0.104)$), but only 3 of 5 players are shared. So per-memory attribution is not unstable; it is *local*: change the workload and 40% of the rows change. With only three overlapping memories we treat this as an observation. It is consistent with the counting limit above.

Table 14: Distinct memories retrieved from a pool of 278 over P=31 episodes. The ceiling column gives kP.

budget	ceiling kP	token overlap	MiniLM	bge-small	MMR
k=3	93	23	36	32	32
k=5	155	38	59	45	52
k=8	248	62	81	65	75
k=16	496	107	127	114	111

Table 15: Shadow fraction and pool lift at different retrieval budgets.

environment	budget	shadow	pool lift (pp)
ALFWorld	k=1	94%	+2.8
ALFWorld	k=10	61%	−8.3
ScienceWorld	k=1 → 10	shadow falls, lift *improves*	

D.11 Game totals and value interpretation

Every attribution game's total value is full-pool success minus empty-pool success. Across the 24 pool/splits with both endpoints, 12 are positive, 7 are exactly zero (`shadow_llama_a/b`, `shadow_off200_a`, `shadow_sci_melt_b`, `shapley_a`, `shapley_llama_b`, `shapley_qwen_b`) and 5 are negative (`shadow_br_a`, `shadow_qwen_a/b`, `shadow_sci_melt_a`, `shapley_sci_melt_a`). On a non-positive-total game, a positive player value means less harmful than the average member of that game, not necessarily helpful in absolute terms. We therefore use rank correlation and separately report the positive-total subset (Appendix D.1).

On the seven zero-total pools the Shapley values are pure interaction terms summing to 0; this is the mechanical reason the band magnitudes on the Llama pools are near zero (mean absolute player values of 0.010 and 0.012), and the reason TMC has nothing to truncate against there, which Table 5 marks. These pools support only an ordering comparison. They also leave limited room for pruning to improve the full game.

D.12 Direct and lineage-mediated scores on natural memory

Section 3 defines indirect influence as the difference between generation and retrieval influence. This section reports the pair on a pool of *agent-written* memory, because the decomposition is the operational output of the generation channel and a single ranking would hide it.

The setting is one failing ALFWorld episode with three retrieved-or-ancestral memories, all written by the agent from its own earlier episodes, scored with the discrete operator of Appendix B.2 using a damping factor of 0.8. Blame is positive when *removing* a memory helps, so a large value means harmful.

The baseline does not score the indirectly implicated memory, while the heuristic indirect component does. This worked case shows that a retrieval-level score can omit an ancestor that appears only through its descendants. The two scores are not a ranking: `48aea935` has the largest *magnitude* of any entry, and a combined score would put it first, but its sign says removing it would hurt the agent. Reporting the pair keeps that distinction visible.

D.13 Pool lift and screening

Replication of the gate on fresh variations. The gate keeps the five ScienceWorld pools with positive lift out of fifteen screened at n=12; ten scored exactly 0.000. Selecting the maximum of fifteen noisy estimates biases the retained lifts upward, so we re-measured the five with the same pool text, retriever, budget and n, on variations 100–111 instead of 0–11 (these tasks have 125–900 variations, so the windows are disjoint).

task pool	screened lift	replicated lift	replicated mem / no-mem
`measure-melting-point-known`	$+0.333$	$\mathbf{+0.000}$	0.083 / 0.083
`inclined-plane-determine-angle`	$+0.333$	$\mathbf{+0.000}$	0.083 / 0.083
`test-conductivity-unknown`	$+0.167$	$+0.167$	0.167 / 0.000
`test-conductivity`	$+0.083$	$\mathbf{+0.000}$	0.000 / 0.000
`lifespan-longest-lived`	$+0.083$	$\mathbf{-0.083}$	0.333 / 0.417
mean	$+0.200$	$\mathbf{+0.017}$	shrinkage $+0.183$

Four of five lifts become zero or negative. This pattern is consistent with selection on noisy $n{=}12$ estimates. We therefore treat the ScienceWorld gate as screening only and the retained pools as having unknown net benefit on fresh variations. The non-positive game totals in Appendix D.11 are consistent with this replication. ALFWorld and τ pools were not selected this way.

On ALFWorld, the screened attribution pool has $v(\text{full}){=}0.429$ and $v(\emptyset){=}0.238$, a lift of $+19.0$pp on the screening tasks.

This is *not* in tension with the non-positive game totals of Appendix D.11, because the two numbers are measured on different sets and we distinguish them throughout. The gate uses the **whole pool** (16 memories on ALFWorld) over the **full task set**; a game total compares only the **player set** (the 4–5 retrieved memories or bands the game is defined on) over **one task split**. The melting-point pool makes the gap concrete: it clears the gate at $+33.3$pp on all 12 variations ($4/12$ successes with memory, $0/12$ without), while its band game on split a records $v(\text{all}){=}0.000$ against $v(\emptyset){=}0.222$. A pool can therefore be helpful as a whole and still present a flat or net-harmful sub-game to the estimator, which is the regime the Shapley values are actually computed in and the reason we score by ordering rather than by magnitude.

D.14 Realised rollout cost

This section reports the number of agent episodes executed in each experimental leg. Table 16 gives the breakdown. These costs are the reason the pools in this paper are small.

Table 16: Realised rollout cost by experimental leg, measured in agent episodes.

leg	episodes	note
utility / pruning (the null)	11,224	most expensive leg; returned a null
exhaustive-coalition Shapley, visible (MemAttr-V)	9,054	15 pool/split attributions
M4 granularity baseline	5,040	the baseline MemAttr-B must beat
M2 item-LOO baseline	2,832	
mechanistic probes (attention, gradient)	2,435	
M1 judge, downstream check	610	
band attribution and lineage targets (MemAttr-B)	492	cheapest attribution leg
cross-split checks	270	
total	**31,957**	

The band estimator uses 492 episodes, compared with 9,054 for the visible estimator. This difference follows from the number of Shapley players. A five-player pool over three seeds and seven evaluation tasks requires 672 episodes; a six-player pool over four seeds and eight tasks requires 2,048. We therefore do not present exact per-memory Shapley as scalable.

D.15 Cross-model robustness

Across ALFWorld pools written by different agent LLMs, item leave-one-out changes its ranking across the fresh runs, while both MemAttr estimators remain positively correlated with their measured targets across the three writer LLMs, although their magnitudes vary. Table 17 reports each method using its native diagnostic, so values across rows are not directly comparable.

Table 17: ALFWorld diagnostics across agent LLMs. Rows use each method's native metric and are not directly comparable across methods. For MEMATTR-V and MEMATTR-B, ρ is Spearman correlation with leave-one-in utility; P@8 chance is 0.33. † denotes results carried over without a Llama rerun.

method	Ministral	Qwen3	Llama3.1
M1 judge (P@8)	0.59/0.69	0.39/0.33	0.59/0.61
M2 item LOO (ranking)	reverses	reverses	(as M/Q)†
M3 gen-DAG (shadow cov.)	100%	100%	(as M/Q)†
M4 granularity (finest level)	2	1	1
M5 binary search (significant?)	no	no	no
MEMATTR-V (ρ)	0.90	0.80/0.94	0.60/0.90
MEMATTR-B (ρ)	0.77/1.0	0.20/0.63	0.83/0.82
$\hookrightarrow$ vs. M4 (sign /8)	**8/8** vs. 1/8	**7/8** vs. 6/8	**7/8** vs. 5/8

D.16 CROSS-SPLIT SOURCE IDENTIFICATION

Summary. With attribution derived on one split and the evaluation target measured on the other, item level loses 2–4 (p=0.69), while band level wins 7–4 (p=0.274). Neither result is significant. On 87 constructed ALFWorld instances with one verified decisive memory, the whole-pool judge reaches $\mathrm{hit@1} = 0.989$ and item LOO reaches 0.759, compared with chance $1/6$. This contrast is consistent with greater ambiguity in natural pools, but does not identify ambiguity as the only cause (Appendix A.6).

Following the downstream protocol used for training-data attribution (Ding et al., 2026) (fix a true source, score every candidate, report the *rank of the true source*) we asked whether our attribution names the memory that mattered. The target is measured, not annotated: the true source on a task split is the memory (or band) with the highest measured leave-one-in value on that split. No new rollouts were needed because all required measurements had already been collected.

Item-level result. Over six cells (3 agent LLMs $\times$ 2 splits) on the 16-memory ALFWorld pool, ranking all 16 memories, MEMATTR-V reaches mean rank 4.9 against 7.2 for a token-overlap retrieval score and 8.5 for both item LOO and chance. The paired result gives a different picture: MEMATTR-V beats retrieval on 2 cells and loses on 4 (two-sided p=0.69). The mean gap comes entirely from two cells where lexical retrieval fell to rank 15, so we do not quote the mean as a positive result. The design is also underpowered. With six paired cells, rejection at α=0.05 requires winning all six, and 80% power would require a true win probability of 0.97. We therefore report a null. Only $\mathrm{hit@3}$ favours MEMATTR-V ($6/6$ vs. $3/6$, p=0.125).

MEMATTR-V assigns exactly 0 to the eleven memories retrieval never surfaced, so an item-level estimator is disadvantaged on a *pool-wide* identification task. It cannot rank memories outside its visible player set.

Band-level result. We repeat the task over generation bands, using the estimator from one split and the measured truth from the other. The tally depends on which pools are counted, so the six independent pools are the primary set. Across these pools (12 cells), MEMATTR-B attains $\mathrm{hit@1} = 5/12$ against M4's $1/12$ and wins the paired comparison 7–4 with 1 tie: **one-sided p=0.274, not significant**.

The alternatives show why the pool definition matters. Restricting the analysis to the four pools used in the paper's rank-correlation comparison gives 6–1 with 1 tie and p=0.062. Adding two non-independent pairs gives 8–4 with 4 ties and p=0.194: `dcfine` re-partitions `dc`, and `bestk` uses the same pool at another retrieval budget. We quote 0.274 because the six-pool set has independent cells and its membership was not chosen after seeing the outcomes. The difference between 0.062 and 0.274 is why we do not present this as a positive result.

At the wider set's effect size, reaching 80% power would require roughly 90 paired cells, about 78 more. Each cell cost about $1{,}100$ agent episodes, so the added experiment would cost about $86{,}000$ episodes, nearly three times the entire paper (Appendix D.14). We report the null together with this cost estimate.

Secondary observations. The measured-best unit survives a change of task split on 1 of 3 pools at item level and 4 of 6 at band level. Second, on the one pool we partitioned both ways, the measured-

best band is stable across splits at four bands and unstable at seven, with the same memories and only the granularity differing. Because the cross-split target is unchanged on the four independent pools that carry the rank-correlation comparison, in-sample and cross-split evaluation coincide there, and the cross-split design cannot separately measure transfer error on those pools.

Third, at item level the same comparison exposes the concrete cost of split-dependence quoted in Appendix A.3: on the Llama pool, MEMATTR-V derived on split b assigns memory 4 a value of -0.049, while measurement on split a makes memory 4 the *most* valuable in the pool at $+0.381$; rank 15 of 16. An estimator can be accurate on the distribution it was derived on and actively misleading one split away.

D.17 THE WHOLE-POOL JUDGE, SCORED TWO WAYS

A whole-pool judge scores $\rho{=}-0.46$ against measured value on τ when ranked over the whole pool and $+0.67$ when restricted to the five Shapley players. The sign changes because the scoring units change. This section reports both conventions in Table 18.

M1 was run in two different output formats. On ALFWorld and ScienceWorld the judge was asked, in Who&When's nomination form, *which* memories helped or hurt, giving a set that admits a precision score but no ranking. On τ-bench it was asked for an *order*, which is why τ was the only environment with an M1 rank correlation. We have since run the ranking form on all three environments with the judge held fixed and scored it under two conventions.

Table 18: M1 whole-pool judge in ranking form. **whole pool** correlates the ranking over every memory with measured leave-one-in utility. **players** restricts both rankings to the 4–5 Shapley players.

environment	judge	split	whole pool	players
ALFWorld	Haiku-4.5	a	$+0.34$	$+0.36$
ALFWorld	Haiku-4.5	b	$+0.38$	$+0.36$
ALFWorld	Sonnet-4.5	a	$+0.26$	$+0.46$
ALFWorld	Sonnet-4.5	b	$+0.19$	$+0.46$
ScienceWorld	Haiku-4.5	a	$+0.35$	$+1.00$
ScienceWorld	Haiku-4.5	b	-0.02	$+0.32$
ScienceWorld	Sonnet-4.5	a	$\mathbf{+0.84}$	$+1.00$
ScienceWorld	Sonnet-4.5	b	-0.07	$+0.32$
τ-airline	Sonnet-4.5	—	$\mathbf{-0.46}$	$\mathbf{+0.67}$

Scoring-unit sensitivity. With one judge held fixed, the whole-pool correlation spans $+0.84$ to -0.46, and restricted to the player set it spans $+0.32$ to $+1.00$. On τ-airline the judge scores $+0.671$ on the same five players and evaluation target on which MEMATTR-V scores $+0.671$. Three of the five target values are tied at $+0.1$, so the correlation is decided by whether one memory reaches the top. At $n{=}4$–5 players these numbers are too coarse to separate methods, so we do not claim that our estimators outperform the judge.

Task-distribution dependence. A whole-pool judge never observes an outcome, so its ranking is a function of the memory text alone. In three of the four environment/judge pairs above the returned ranking is *bit-identical* across our two disjoint task splits, while the measured value is not: on ScienceWorld the same Sonnet ranking scores $+0.84$ against split a and -0.07 against split b. A judge *shown only the memory text* therefore cannot be split-stable even in principle. A judge also shown representative tasks, a workload description, or past outcomes is outside this analysis and remains a baseline for future work. Our result applies to the text-only whole-pool judge: because its score is independent of the task distribution, it is not an estimator of task-dependent causal value. This is the granularity argument of Appendix A.3 in its limiting case, an estimator whose output cannot depend on the distribution the estimand is defined over.

Comparability across player sets. Scoring one method over 12 memories and another over 5 players in the same column can reverse the sign of a headline number, which is why we state the player set with every ρ we report.

D.18 Power of the pre-registered test

Scope of the pruning test. The intervention tests deletion at a fixed budget. At trace level, Zhang et al. (2026b) convert an attribution into $+4.8$–14.2% by *feeding the diagnosis back*, and reflection baselines (Madaan et al., 2023; Gou et al., 2024) *degrade* performance by up to 5.5% in their settings. We did not test feedback. Half our pool/splits carry a non-positive game total to begin with (§5), bounding what pruning could recover from them. Empty-prune arms measure a floor of $+0.054$ on ALFWorld but only $+0.000$/$+0.013$ on τ-retail, so the floor is environment-specific and on τ-retail it does *not* explain away P5's advantage over P3 (Appendix C.6).

We report the minimum detectable effect of the confirmatory design of §6.3.

Significance of the observed reversal. Under mean-based task resolution, 12 of the 40 tasks are discordant and P3 wins 1, which is two-sided p=0.0063 with a 95% interval on P3's conditional win probability of $[0.002,\ 0.385]$. That number depends on the resolution rule: it resolves each task to its *mean* over the four seeds and counts any difference of means as a win, which lets a $2/4$ versus $3/4$ task contribute a full discordant pair. Resolving each task instead by majority vote, the reading consistent with our own argument that the statistical unit is the task and not the (task, seed) cell (§6), leaves only 2 discordant pairs, P3 winning 0, two-sided p=0.50. The *direction* is against P3 under both resolutions, and every arm-level estimate is against it, which is what §6.3 claims. We report both resolutions because the significance depends on this choice.

The design is a one-sided sign test on paired per-task success over 40 held-out τ-retail tasks, at α=0.05. A sign test sees only *discordant* pairs, and 12 of the 40 tasks were discordant under mean resolution, so the effective sample size is 12. The test rejects only if P3 wins at least 10 of those 12 tasks. Reaching 80% power requires P3 to win 87% of discordant tasks, about 6.7 times as often as P5. At the observed 30% discordance rate, this corresponds to a pooled success-rate gap of roughly 22 percentage points. Table 19 gives the same computation at higher discordance rates, which is the axis that would increase power here: even if *every* task were discordant, 40 pairs would only resolve a $2.3\times$ win ratio.

Table 19: Minimum detectable effect of the confirmatory sign test at one-sided α=0.05 and 80% power, as a function of the number of discordant task pairs. The observed row is m=12.

discordant pairs	reject if P3 wins	MDE: P3 win probability	implied win ratio
12 *(observed)*	≥ 10	0.87	$6.7\times$
20	≥ 15	0.80	$4.0\times$
40	≥ 26	0.70	$2.3\times$

The failed one-sided test is weak evidence. The two-sided reversal statistic also depends on the task-resolution rule. Under both rules, P3 loses to the size-matched control on both splits at the arm level (-0.075/-0.050 against $+0.062$/$+0.000$) and wins a minority of discordant tasks. Detecting a small advantage would require more tasks or a higher discordance rate.

E Extended discussion

E.1 Limitations

The evaluation target is leave-one-in utility, which is neither estimator's estimand. The measured target is the success added by a memory or band when inserted alone into an empty pool. It is the singleton stratum of the Shapley expansion rather than I_{ret}, I_{gen}, or the Shapley value itself. It is what we can measure at 4–7 players without a full coalition table per cell, and it correlates with the Shapley value by construction on games close to additive, which biases agreement upward in the regime where attribution is easiest. Agreement with this target does not establish recovery of I_{gen}. In the one direct measurement, band Shapley tracks I_{gen} at only ρ=+0.32 (Appendix D.7).

The two shadows use different intervention sets. The unreachable shadow in Def. 3.2 depends on the sub-pools an estimator tests. The observed shadow is the never-retrieved set on our sampled tasks. These sets need not be equal (Appendix A.2).

The main tables are within-split. Table 4 and Table 17 score estimator and target on the same task split; the cross-split form of the same question is not significant at our sample size (p=0.69 at item level, p=0.274 at band level, §6.2), and the canonical band partition is not what every pool uses. Three pools (`dc`, `dcfine`, `bestk`) are partitioned by lineage *level* rather than by source, a departure from the construction in Appendix A.5 we keep only because it is what makes the granularity comparison controlled (Appendix A.11).

Scale. Pools are 8–16 memories where rollouts are needed and 278 where they are not, over 4–40 evaluation tasks, so several results are directional and we mark them instead of bolding them; the zero-rollout measurements (§4) are the best-powered in the paper because they are the ones we can run at scale.

One semi-natural environment. SWE-bench memory is gold-patch-guided because the fully natural version gave floor performance on stably-failing tasks; we disclose it in Table 4 instead of pooling it.

The null is single-domain. The pre-registered replication ran on τ-retail only, because no unused task ids remained in τ-airline.

Probes use open-weight proxies. Attention and gradient signals require weights, so Figure 4 uses 7B open models, the judge result does use the deployed agents.

The retrieval curves are descriptive. Power and logarithmic curves both fit the measured grid, so we cannot tell which curve is correct and do not use either one beyond the observed pool sizes (Appendix A.13).

The lineage weight is validated on constructed scenarios, and is not the partition we deploy. The stance-gated lineage weight is a modelling choice; Appendix B.3 validates it on controlled parent sets, not on a large sample of naturally-written children. Its gate is an LLM call, though no longer an irreproducible one: three judges including an open-weight local model agree on every constructed edge and on 93% of the gate-relevant decisions on a real DAG, and three embedders agree on every ranking (Appendix A.10). Applied to our own logged DAGs, the gate keeps only 18% of edges and promotes 75% of children to sources. The resulting partition agrees with the one used in our results on 6 of 20 memories. The band results therefore use the source-rooted partition of the logged DAG, and the stance gate is a validated component we do not deploy (Appendix A.9). It also measures stance inheritance rather than ablation-style harm, and we do not claim the two coincide.

Bands need a logged lineage. Band-level attribution presumes the generation DAG was recorded, and agent frameworks do not all log which memories were in context when a memory was written, which we regard as a concrete recommendation for memory implementations, not merely a limitation of ours.

E.2 Additional related work

Training-data attribution. Valuing training examples faces a *computational* obstacle rather than an informational one: exact Shapley needs retraining per subset (Shapley, 1953; Ghorbani & Zou, 2019; Lundberg & Lee, 2017), so the literature is a sequence of approximations, from influence functions at small (Koh & Liang, 2017) and large (Grosse et al., 2023) scale to in-run estimators (Ding et al., 2026), aggregation-sensitive variants for language models (Vitel & Chhabra, 2026), and extensions to meta-learning (Ren et al., 2025). Our obstacle is missing retrieval support, which is why we change the unit rather than only the estimator. This literature is a neighbour rather than a baseline because none of its estimators is defined on a memory the agent never retrieved.

Coalition-structure values. Taking Shapley over a *partition* rather than over individuals makes the coalition-structure value the standard object (Owen, 1977). Our band value is that object, not an aggregate of item values, so we report the two quantities separately.

Relation to in-run data attribution. The closest methodological neighbour is In-Run Data Shapley (Ding et al., 2026), which makes attribution tractable along a *single realized training trajectory* by defining a *fixed-state* local utility, holding the model state fixed while a datum's marginal contri-

bution is evaluated, and then restoring an efficiently aggregable dot-product form with a linearised approximation of the update map.

First, the shared word "additive" means different things. Theirs is a statement about *computation*: the linearised form lets per-datum contributions be aggregated as dot products instead of materialising per-sample gradients. Ours is a statement about the *estimand*: whether per-item credits sum to the effect of the whole pool, which we measure failing, the summed item credit reaching 167% of the pool effect (Appendix A.3). Their device does not assume our property and our measurement does not contradict their device.

Second, the two settings address different problems. In-Run Data Shapley faces a computational difficulty and answers it with a tractable surrogate along the realized trajectory. Our evaluated retrieval logs contain no item-level support for much of the memory pool, so we change the unit. Fixing model state makes their utility tractable where the model *state* varies between counterfactuals; grouping memories into bands yields a defined counterfactual where the retrieved *context* varies. We also use their downstream protocol directly, scoring the rank of a true source among candidates (Appendix D.16).

Agent failure attribution asks which step or agent caused a failure (Zhang et al., 2025), typically with an LLM judge over a trace. Zhang et al. (2026b) advance that line in three ways that bear directly on us. Their results are consistent with our judge finding: state-of-the-art reasoning models attribute failure at *below* 10% accuracy on their setting, the step-level counterpart of the judge unreliability we measure over pools. Their evaluation target is created by counterfactual replay with programmatic fault injection, whereas ours is measured leave-one-in on agent-written memories with no planted faults. And they convert attribution into downstream gain by *feedback*, not deletion (§6.3). Their benchmark is also far larger than ours ($2{,}000+$ curated trajectories against our pools of 8–278 memories); we do not compete on scale, and what MemAttr-Bench adds instead is the *generation lineage* and per-band measured evaluation targets, neither of which a trace corpus contains.

Memory architectures and memory benchmarks. Zhang et al. (2026a) replace retrieval over discrete texts with generative latent memory, which places their setting outside our formalism (§3): there is no retrieved set, so there is no per-item counterfactual to estimate. On the evaluation side, Hu et al. (2026) benchmark memory *capability* along four competencies (accurate retrieval, test-time learning, long-range understanding, conflict resolution). That is an orthogonal axis to ours: they ask whether an agent's memory works, we ask whether the credit for its working can be assigned, and a system could score well on one and be entirely unattributable on the other. We port that method family from traces to memory pools (§5) and find the judge transfers badly, though not in the direction we first reported: its ranking cannot depend on the task distribution at all, so it is unreliable rather than inverted (Appendix D.17), consistent with wider evidence that judge signals are fragile (Li et al., 2026a).

Self-improving and memory-augmented agents write and reuse their own experience (Shinn et al., 2023; Wang et al., 2023; Park et al., 2023), and the loop can degrade behaviour: Shao et al. (2026) document misevolution including safety drift after memory accumulation. That establishes the failure exists; we ask which memories existing estimators can score and measure that retrieval does not expose most of the pool. Where related theory studies what accumulating context does to a *learner* (Li et al., 2026b), we study how much of that context attribution methods observe.

Retrieval and context attribution explains an output through the items in its prompt. By construction, it does not score memories outside that retrieved context.