RETRIEVE AND COMPRESS: WHY NAIVE EXPERIENCE SHARING FAILS FOR WEB AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Giving a web agent access to other agents' past trajectories is widely assumed to help. We show that unselective sharing does the opposite. On WebArena-Verified (812 tasks, 5 seeds, one independent memory pool per seed), injecting every accumulated trajectory into every subsequent task *drops* success from 30.0% to 12.6%, below an agent with no memory at all, as does a curated pool read at random (-9.9 pp)—in neither case is what gets injected chosen for the task in front of it. What recovers the gain is not sharing but discipline: deciding what may enter the pool, what survives compression, and which entry a task is shown. That reaches 41.4% against a 30.0% baseline (+11.4 pp, task-level paired $p < 0.005$, Holm-adjusted across the 13 memory conditions)—real but modest, Cohen's $h = 0.24$. Of those three decisions only one changes the *sign*: that design's own pool, read without the intent rule, scores 20.1%—+21.3 pp lower, below the no-memory baseline—while removing compression or admission leaves it above. The direction holds across task type, task difficulty and model family, and more weakly on a second benchmark (Appendix C.10). An oracle injecting each task's best historical trajectory reaches 53.4%, +12.0 pp above us—an upper bound, not headroom—and no configuration of ours approaches the strongest specialised systems on the benchmark.

1 INTRODUCTION

A web agent that solves a task produces a trajectory: a sequence of observations and actions that, for at least some tasks, actually worked. When many agents run against the same sites those trajectories pile up, and the natural move is to let later agents see them. This is the intuition behind experience reuse for language agents (Zhao et al., 2024; Shinn et al., 2023), and intuitive enough that the sharing step is usually assumed beneficial rather than examined.

We examine it. Our first result is negative and, we think, the most useful thing in the paper: **sharing unselectively is not merely unhelpful; it is worse than having no memory.** Running all 812 WebArena-Verified tasks with a pool that admits every finished trajectory and injects all of it yields 12.6% success, against 30.0% for an agent that never sees another agent's work; a curated pool read at random lands at 20.1%. The failure is measurable in four ways, none of them the success rate itself (Section 5)—so a report giving only scores would hide it.

Memory helps only when it is *curated*, *compressed* and *retrieved*; our second contribution separates those three factors rather than bundling them. All three are load-bearing: removing the retrieval step costs +21.3 pp, more than the entire gain; switching off the admission rule costs +8.6 pp; and removing compression costs +7.8 pp (+4.0 on the payload, +3.8 on the index at most), most of it.

Our third contribution characterises the boundary of the effect rather than asserting it is general: the gain is +17.9 pp on navigation, +13.2 pp on retrieval and +7.8 pp on computation, and +13.8 pp on the hard subset against +10.3 pp on the rest—what one expects if memory shortcuts *how to get somewhere* and *where something is* but not *what to compute once you are there*.

Finally we bound our own claim: an oracle that injects each task's best *historical* trajectory reaches 53.4%, +12.0 pp above us, and we say what that does and does not measure rather than calling it headroom (§5). The direction also holds for a second model family (+10.7 pp with GPT-5-mini against

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.

+13.8 pp for our main model), and no configuration of ours approaches the strongest specialised systems on this benchmark.

2 Related Work

2.1 Web agents and their evaluation

WebArena (Zhou et al., 2024b) provides realistic self-hosted web tasks and remains the standard evaluation; its original string-match evaluator was later shown to produce substantial false negatives, which WebArena-Verified (El hattami et al., 2025) replaces with human-audited deterministic checkers. We use the verified evaluator throughout and store its version and checksums with every result record. Neighbouring benchmarks trade reproducibility for coverage: Mind2Web (Deng et al., 2023) evaluates on live public websites online and on cached snapshots offline, and WorkArena (Drouin et al., 2024) on an enterprise knowledge-work platform. We replay the Mind2Web offline splits as a transfer check rather than as a second benchmark (Appendix C.10), and report nothing on any other environment; whether our conclusions generalise past one task family is therefore still open. That is a gap the benchmark side has begun to close: MemGym (Xu et al., 2026) unifies tool-use, search, coding and computer-use tasks behind one memory interface and scores each strategy with the reasoner held fixed, the design our transfer layer approximates and the evaluation we would need to run to say the effect is not specific to web tasks. Agent architectures for the web build on ReAct-style interleaved reasoning (Yao et al., 2023; Wei et al., 2022); we keep our scaffold unoptimised so any measured difference is attributable to memory rather than prompt engineering; a stronger grounded perception stage (Zheng et al., 2024a) would raise every row in our tables rather than reorder them.

2.2 Experience reuse within one agent

Reflexion (Shinn et al., 2023) converts failures into verbal self-reflections; ExpeL (Zhao et al., 2024) extracts reusable rules from cross-task experience; Agent Workflow Memory (AWM) (Wang et al., 2025) induces reusable workflows from past trajectories and reports a large relative gain on WebArena, the closest published result to ours and the one our calibration table anchors against; Voyager (Wang et al., 2024) keeps a growing skill library; MEM1 (Zhou et al., 2025) carries memory and reasoning together in a fixed-size state. These systems operate within a single agent's own history, and the recent ones no longer treat accumulation as sufficient: Contextual Experience Replay (Liu et al., 2025b) synthesises past experience into a buffer of environment dynamics and decision patterns, ReasoningBank (Ouyang et al., 2025) distils strategies from self-judged successes *and* failures, Agent S (Agashe et al., 2025) augments hierarchical planning with retrieved experience, and WebCoach (Liu et al., 2025a) keeps cross-session memory and injects retrieved advice at runtime—each choosing what reaches the prompt rather than injecting what accumulated, and ReasoningBank reporting that this beats storing raw trajectories or successful routines alone. Agent KB (Tang et al., 2025) shares a knowledge base across heterogeneous frameworks behind a retrieval gate; we read its gains as consistent with our selection result rather than as a counterexample to the negative one, since it too retrieves selectively. None of them sweeps the shared artifact itself, so how much of it to keep is a choice each makes rather than a quantity anyone has measured. We hold the protocol fixed and vary admission, storage and selection as the treatment, whole-pool injection being one point on that grid, and the Synapse-style baseline of §4 is our own port of it (Zheng et al., 2024b).

2.3 Memory and replay: the same question elsewhere

That more experience is not automatically better is established outside language agents: prioritized replay samples transitions by estimated usefulness rather than uniformly (Schaul et al., 2016), replaying other agents' experience is off-policy enough to destabilise learning unless corrected (Foerster et al., 2017). In language agents the precedents sit in other settings: expanding an agent's accessible history erodes cooperative intent in multi-agent games, through the agent's reasoning rather than through prompt length (Liu et al., 2026), and an exemplar being a correct instance of its task is no guarantee that it helps (Qiu et al., 2026). Our result is an instance of the same phenomenon in web-agent task success, with the policy held fixed. Generative Agents (Park et al., 2023) and MemoryBank (Zhong et al., 2024) treat retrieval as a first-class decision, which we vary rather than

assume; a recent study varies the memory itself inside one agent and measures what it follows (Xiong et al., 2026).

2.4 Compression and retrieval

Variational autoencoders with a β-weighted KL term (Higgins et al., 2017) provide the information bottleneck we use to compress trajectories; sentence embeddings from a frozen encoder (Reimers & Gurevych, 2019) provide the retrieval key, and the similarity index we search is the small-pool case of standard approximate nearest-neighbour tooling. Retrieval-augmented generation (Lewis et al., 2020) supplies the general framing—retrieval quality bounds downstream quality—and our oracle condition probes that bound for this task rather than assuming it, though it reads each task's own history and so over-states what retrieval alone could reach (§5). Granularity is a design choice in its own right: TRAD (Zhou et al., 2024a) retrieves per step, on the grounds that a whole trajectory can be plausible yet state-agnostic, and we retrieve whole trajectories once, at the first step, leaving that failure mode open by construction. Agent memory is usually compressed by a language-model summariser. We set the compression rate explicitly (β, latent width) and render the payload from it deterministically (§3), so the rate is a design parameter rather than a property of a summarisation prompt, and §6.3 reports the sweeps over it; for the wider design space see the survey of Luo et al. (2026).

2.5 Multi-agent systems

A parallel literature has agents collaborate by exchanging messages: AutoGen (Wu et al., 2023), MetaGPT (Hong et al., 2024), CAMEL (Li et al., 2023) and scaling studies of group size (Qian et al., 2025); see also the survey of Xi et al. (2025). In these systems the shared object is a conversation, consumed within an episode and discarded; we study a persistent shared *artifact* that outlives its authors, so its size and composition are design variables that can be set badly enough to make it worse than none.

3 Method

3.1 Setting

Three agents per seed solve a stream of web tasks in a randomised order, sharing that seed's pool. Each is an unmodified ReAct-style loop with a fixed budget of 15 environment steps, a protocol constant across every condition and not the token budget, writing finished trajectories to the pool and, before acting, reading from it. The pool is *per seed*: seeds never share state, and the pool directory carries its seed, so cross-seed reuse is not merely discouraged but unrepresentable. This is the paper's most important design decision; §4.6 explains why.

3.2 Curation

A finished trajectory is admitted only if its task succeeded *and* it is not a near-duplicate of an entry already in the pool, where duplication is cosine similarity above 0.95 in the storage space. Two pool properties are reported beside that rule rather than as counts of what it turned away: the duplicate rate is the share of pool entries that near-duplicate another, and the misleading share is the share the annotation pass judged misleading on injection—content that would lead the agent to a wrong page or answer. Neither is a rejection count, so both are defined for an uncurated pool too, and neither is mechanically lowered by admitting more entries.

3.3 Bottleneck

An admitted trajectory is encoded by a frozen sentence encoder (`all-MiniLM-L6-v2`, 384-d) and passed through a β-VAE ($384 \to 256 \to \{\mu, \log \sigma^2\} \to d$) with $d = 128$ for our main configuration. The VAE is trained per seed on that seed's own pool, so no seed's representation depends on another seed's data. We log the reconstruction fidelity—cosine between $\mathrm{decode}(\mathrm{encode}(x))$ and x, averaged over the pool—as a diagnostic rather than a claim that the bottleneck preserves information: a high value is reconstruction in the representation's own metric, not a licence to call the payload lossless, and says nothing about whether another width or β would do better (§6.3; constants in Appendix A.7).

We are explicit about what the bottleneck is and is not. It is the *retrieval index space* and the duplicate-detection space. It is not a text generator: a 128-dimensional latent does not decode to English, so the compressed payload injected into the prompt is not a decode of the latent. That payload is the output of a deterministic extractive renderer—a pure function of the trajectory text, with no model call—which keeps the salient actions and short observation excerpts under a character budget set by the latent width (Appendix A.7). A stronger reading would require a sequence decoder, which we do not have and do not claim.

3.4 Retrieval and injection

At a task's first step—before any observation of its own—the agent queries from the task intent and retrieves the $k = 3$ pool entries with highest cosine similarity in the space the pool is stored in. Entries recorded for the querying task itself are excluded from the candidate set. The query is that intent's 384-d sentence embedding passed through the same encoder as the entries, taking the posterior mean rather than a sample, so a query and an entry are compared in one space; the encoder was trained on pool trajectories and is applied here to task intents. Retrieved payloads go into a bounded "shared experience" block, subject to deterministic first-in-first-out eviction if the context window would otherwise overflow; every eviction is logged.

4 Experimental Setup

4.1 Benchmark

All 812 tasks of WebArena-Verified, scored by its deterministic human-audited evaluators. Partial credit does not exist, so each task is one Bernoulli trial and one episode maps to one trial. Subsets are slices of the same runs, not separate runs: the hard subset and the breakdowns in Figure 3 are cut from the main-layer data. The hard subset is the file shipped with the benchmark release we pinned—`webarena-verified-hard.json`, checksum `d20872f9. . . b14e68`—which in that version contains 258 task ids; we identify it by file and checksum rather than by task count: the count belongs to the release rather than to the subset, and published descriptions have quoted a smaller one. The non-hard complement is the remaining 554. Appendix C.10 adds a transfer check on a second benchmark, with its own task lists and its own runs; none of it is part of this task set.

4.2 Task composition

The 812 tasks span six sites and three task types (navigation, retrieval, mutation). Type by site is not, and cannot be, perfectly crossed: Reddit has no navigation tasks and Map and Wikipedia no computation tasks, so three of the eighteen type-by-site cells are empty in the benchmark itself and a balanced design is structurally unavailable rather than merely inconvenient. We maximise the smallest cell and report the cross-tabulation we actually achieve (Appendix A.5) rather than claiming a crossed one, and we prefer the difficulty split when a comparison must be free of the type/site confound. The per-type numbers in Figure 3 are descriptive, not factorial.

4.3 Models

DeepSeek V4-Flash is the main model, run over the full task set for every condition. GPT-5-mini is used only for the cross-model check (hard subset, conditions A and D). Vendor names are not version identifiers—a name can be re-pointed at a newer snapshot, or retired mid-study—so we record the identifier the endpoint returns for every batch and abort on a mismatch against the first, making a re-run against a different snapshot visibly a different run rather than a correction to this one.

4.4 Conditions

The design is a three-axis grid—*storage* (raw trajectory / VAE-128 / key-point summary), *selection* (none / all / random / intent similarity / oracle) and *curation* (success-only / all / success-plus-dedup)—rather than a set of code paths, and the 16 conditions of Table 1 are read against it rather than being points on it: eight are grid points, four are controls that isolate one thing, three are other authors' systems re-implemented on that pipeline, and two—A, which has no memory, and D′, which changes

only the payload D injects—sit off it. A further point is reported without a table row: G widens the bottleneck to 256 dimensions at D's selection and curation, solving 320.6 of 812 tasks (§6.3). The controls are A+tok, A given a per-episode ceiling of $8{,}000$ tokens instead of D's budget; Filler, A given that budget in task-irrelevant text; C, F and Rand-proj, which replace one of D's choices—its selection rule, or the projection that trains the index—with a random one; No-admit, which is D with the admission rule switched off and everything else held; and D′ and E, which separate the payload form from the representation at a fixed budget, though E drops deduplication (§5.8). We additionally report an oracle retriever, causal controls that keep D's pipeline but substitute random or failed memories for the retrieved ones, and re-implementations of three published memory mechanisms: Reflexion (Shinn et al., 2023); a trajectory-as-exemplar memory in the style of Synapse (Zheng et al., 2024b); and rule extraction in the style of Expel (Zhao et al., 2024). Our Synapse port keeps the published retrieval key and draws from D's curated pool rather than from hand-supplied demonstrations, so we read it as a floor on that method under our scaffold rather than as a reproduction of its numbers.

4.5 Metrics

Success is necessary but far from sufficient, so every task execution records cost, retrieval and pool diagnostics alongside its score; Appendix A lists each quantity and what it means. Reported token counts separate the browsing loop from the final answer-formatting call, memory-free in every condition, so the memory effect is measured on the browsing policy alone.

4.6 Statistics

Comparisons are task-level paired McNemar tests on the 812 tasks, the unit of randomisation, so that agent- or seed-level aggregation cannot inflate the effective sample size. Three agents and 5 seeds run every task, so replicates are collapsed by majority over the three agents—one binary outcome per task per seed—and the 812×5 pairs are pooled into one McNemar test ($b = 1087$, $c = 625$, $p \approx 10^{-29}$). Pooling treats repeat runs of one task as independent, which the intervals below do not, and the weakest of the 5 per-seed tests is $p < 10^{-4}$; it is not the primary test, because the maximum of 5 dependent p-values has no uniform null. Multiplicity across the condition grid is controlled with Holm's procedure; the bound we quote is the family's largest adjusted value, set by No-admit, and every other condition clears it (Table 2). We report Cohen's h so that a small p-value from a large N is not mistaken for a large effect.

Intervals come from the paired and cluster bootstraps defined in Appendix A, which resample tasks and whole intent templates respectively. Resampling at the template level is what widens the interval, by a design effect of 1.7 (an intra-template correlation of 0.1), and the widening is the clustering rather than noise, so we quote the cluster interval where the two differ and no draw reverses the sign ($p < 0.0001$). Every claim below is a paired difference on the same tasks, not a gap between two independent rates, and the resolution a single condition is read at is ± 3 pp (Appendix A), so a 2–3 pp gap between rows of Table 1 is not a finding.

5 Results

5.1 Sharing without selection is worse than no memory

Table 1 gives the main layer. The headline is the contrast between A and B: an agent that reads everything every other agent has written scores 12.6% against 30.0% for an agent that reads nothing. The pool is not merely unhelpful, it is harmful, and the cost table says why—B's prompts average 18.6k tokens, its agents hit the step ceiling more often (15.0 steps on average against 13.2 for A and 9.6 for D), it is the only condition that overflows its context window (47 evictions, against 3 for E and none elsewhere), and only 21% of what it injects is judged relevant—a share of everything it stores, against rates over a retrieved subset where retrieval exists (Table 5). An information-overload account predicts all four at once. It does not explain C and F: a curated pool, no overload, still below the baseline.

Table 1: Main layer: all 812 tasks, DeepSeek V4-Flash, 5 seeds, an independent pool per seed. Success is the mean over seeds ± its SD; *Solved* is the mean task count behind the rate, so dividing it by 812 recovers the rate; Δ is the paired same-task difference against A, seed by seed, and equals its rate minus A's. Per-seed counts: Table 6.

Group	Storage → Selection	Success (%)	Solved	Δ vs A (pp)	Role
baseline (A)	— → —	30.0 ± 1.6	243.8	—	no memory
capped context (A+tok)	— → —	31.7 ± 1.7	257.6	+1.7	context-ceiling control
length matched (Filler)	— → —	29.6 ± 1.7	240.6	−0.4	length-matched control
share all (B)	raw → all	12.6 ± 1.5	102.6	−17.4	information overload
random select (C)	raw → random	18.4 ± 1.6	149.6	−11.6	worse than no memory at all
ours (D)	VAE-128 → intent sim.	41.4 ± 1.6	336.2	**+11.4**	**the full design**
ours, no admission (No-admit)	VAE-128 → intent sim.	32.8 ± 1.6	266.2	+2.8	admission removed
ours, raw payload (D′)	VAE-128 → intent sim.	37.4 ± 1.7	303.6	+7.4	payload form isolated
raw storage (E)	raw → intent sim.	33.6 ± 1.7	272.8	+3.6	representation ($D' > E$)
ours, random proj. (Rand-proj)	rand-128 → intent sim.	35.9 ± 1.6	291.8	+5.9	random projection, same width
ours, random select (F)	VAE-128 → random	20.1 ± 1.6	163.2	−9.9	retrieval needed, causal floor
causal control (CF-failed)	VAE-128 → intent sim.	15.8 ± 1.6	128.2	−14.2	cost of misleading memory
oracle (Oracle)	raw → oracle	53.4 ± 1.9	434.0	+23.4	retrieval upper bound, one entry
Reflexion	summary → recent	36.3 ± 1.7	294.6	+6.3	verbal self-reflection
Synapse	raw → similarity	34.6 ± 1.7	280.8	+4.6	trajectory memory, similarity key
Expel	rules → similarity	34.2 ± 1.7	277.6	+4.2	distilled rules, similarity key

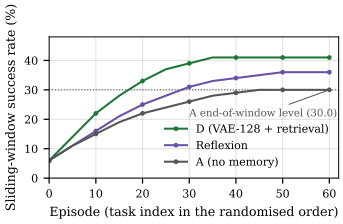

Figure 1: Sliding-window success rate against episode, an episode being a task index in that seed's randomised order. The three curves are A (no memory), D and our Reflexion re-implementation. D passes A's end-of-window level—the dotted line—between episodes 15 and 20, and is flat from episode 35 onward. The axis stops at 60 episodes, so post-60 behaviour is untested.

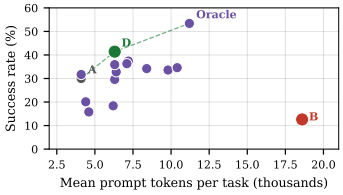

Figure 2: Mean prompt tokens per task against success rate, one point per condition; B, the widest and worst, is drawn in red. The dashed line is the frontier once the token-matched control is excluded—A, D and the oracle. D sits on it at 6.3k, above every cheaper condition, with only the oracle higher. A+tok is excluded as a manipulation check: at A's own 4.1k it buys 31.7%, so it would otherwise be the leftmost vertex.

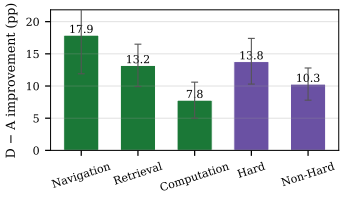

Figure 3: Improvement over baseline by task slice: five bars, three task types and two difficulty slices. Each bar is a marginal contrast within its slice, from the same 812-task runs, not a re-run; weighting either partition back by task counts returns the headline +11.4 pp. All five intervals exclude zero, and within a partition narrow as the slice grows: among the types, computation's 374 tasks are narrowest (+5.0 to +10.6), navigation's 113 widest (+11.9 to +23.9). The type split is confounded with site (§4).

5.2 The gain is not bought with compute

The cost table (Appendix 8) points the opposite way from the usual expectation. D takes 27% fewer steps than the no-memory baseline (9.6 against 13.2), finishes faster (102s against 128s), and in Figure 2 sits on the cost–performance frontier, scoring above every condition that costs less than it does. It does *not* do so on tokens: its prompt is 6.3k against A's 4.1k, about 1.5× longer, so the advantage is in fewer, better-directed steps rather than a smaller context. The comparison that matters here is against B, which spends the study's *highest* token count (18.6k) for the *lowest* score; more generally, accuracy and cost do not order the conditions the same way (33.6% at 9.8k sits below 36.3% at 7.1k): there is no single cost axis along which the full design wins.

5.3 Selection decides the sign; admission and compression the margin

D reaches 41.4% (+11.4 pp over A). The gain accrues and then levels off from episode 35 (Figure 1). The difference is significant at the task level (McNemar, Holm-adjusted $p < 0.005$ over the 13

comparisons against A) and robust to the clustering of tasks within intent templates: a paired bootstrap over tasks gives +9.3 to +13.4 pp and a cluster bootstrap over intent templates +8.2 to +13.6 pp. Cohen's $h = 0.24$ is reported deliberately: that is a *small* effect, and it is the number to carry away rather than the p-value, since with 812 paired tasks a small shift reaches significance.

Removing similarity selection while keeping the same storage—condition F—costs +21.3 pp, putting it -9.9 pp below the baseline. The pool is still D's, so that gap is attributable to *which* memories are injected rather than to injecting: the comparison holds the mechanism, the prompt template and payload construction fixed and changes only the selection. It does not hold the length fixed—unrelated trajectories render to shorter payloads, 4.4k against D's 6.3k—a confound on this contrast and not the ones around it. Removing the bottleneck while keeping similarity selection—condition E—costs +7.8 pp, leaving it +3.6 pp above that baseline. Removing the admission rule—condition No-admit, which keeps D's storage, key, payload and k and admits every finished trajectory—costs +8.6 pp (+6.7 to +10.6) and leaves it at 32.8%, still above the no-memory baseline, so curation is load-bearing but not sign-critical. Only one of the three decides the sign. C, selecting randomly from raw storage, is below the baseline at 18.4%. D′ separates the two things E holds together: it keeps D's index and retrieval but injects the payload uncompressed, so D−D′ (+4.0 pp) is what compressing the payload is worth and D′−E (+3.8 pp) at most what compressing the index is, since E also admits duplicates; the two account for the whole +7.8 pp between D and E. Widening the bottleneck (G, 256 dimensions, D's selection and curation) leaves the rate at 39.5% ± 1.8, +9.5 pp over A and within D's resolution, so the index's width is not where the gain is decided (§6.3). Nor is learning the whole of it: a fixed random projection of the same width, trained on nothing, reaches 35.9%, +5.9 pp over the baseline, and training the projection is worth +5.5 pp beyond that.

5.4 The gain is not a longer context

Filling A's prompt with task-irrelevant text at D's own budget (6.3k) leaves it at 29.6% against A's 30.0%, an interval containing zero (-2.2 to +1.5): spending D's budget on the wrong content buys nothing. A+tok, which gives A a ceiling rather than a spend, says the same thing from the other side—31.7% at A's own 4.1k, +1.7 pp over A and far from D. The benefit comes from the content of the retrieved experience, not from more room to think.

5.5 Misleading memories cause measurable damage

CF-failed keeps D's pipeline but injects failed trajectories instead of successful ones and lands at 15.8%—below D and below even the no-memory baseline. Failure is not merely uninformative here; it is contagious—the strongest argument that curation is a separate design decision, not a by-product of compression. Appendix 4 shows the same effect dynamically, with the misleading share climbing monotonically wherever failures are admitted.

5.6 External memory baselines

Under our protocol Reflexion reaches 36.3% (+6.3 pp over A), the trajectory memory 34.6% (+4.6 pp), and rule extraction 34.2% (+4.2 pp). Each is tested against A on the same tasks, not against the ±3 pp band a single condition carries: all three paired intervals exclude zero, at +4.4, +2.7 and +2.2 pp, so all three are gains, and Reflexion—explicit self-critique—remains the closest any baseline comes to us. None is close to D, and we report all three as re-implementations under our scaffold rather than as the published systems' numbers. The three intervals overlap, so we rank none of them.

5.7 The direction is not contradicted by a second model

Table 4 (Appendix C) repeats the A/D contrast on the 258-task hard subset with a second model family. GPT-5-mini shows a +10.7 pp gain against +13.8 pp for DeepSeek V4-Flash on the same tasks, and both rows carry the same paired bootstrap over tasks. The two intervals overlap heavily—each contains the other's point estimate—so this comparison does not separate the models; the evidence it offers is that the direction is not an artefact of one model's prompting quirks.

5.8 Ablating without confounding

Table 7 (Appendix C) varies retrieval key and storage representation directly; its limits come first. The column axis is clean, but the raw×intent cell is E, which admits duplicates: within a row the payload, pool and prompt template are fixed and only the key changes. The row axis moves representation and payload together—raw entries in full, VAE-128 and summary as reconstructed text—so a row difference is not a compression contrast; the compression claim rests on D′, where the payload is held uncompressed and only the index changes, an upper bound (§5). Three things hold regardless. First, the compressed representation is the best row under every key, but the margin over the runner-up is comfortable only under the intent key (3.9 pp) and thin under the other two (1.8 and 1.6 pp, both inside the ± 3 pp resolution of §4). Second, the best cell is VAE-128 with the intent key—the D configuration, 41.4%—and the worst is raw trajectories with the page-state key at 30.2%, where the no-memory baseline sits (30.0%). Third, against our own interest, the two axes are of comparable size: spreads across keys within a row of 4.9, 6.5 and 5.1 pp, against 7.8, 4.7 and 4.7 pp across representations within a column. We cannot claim the representation matters more than the key, and with three points per axis we offer both orderings as patterns rather than findings. Row and column effects are additive to within 1.2 pp at the worst cell; only the spreads above are read as signal.

5.9 Where memory does not help

Figure 3 is the part a practitioner should read. The improvement is +17.9 pp on navigation and +13.2 pp on retrieval tasks but only +7.8 pp on computation, and +13.8 pp on hard tasks against +10.3 pp on the rest. A shared trajectory tells a later agent where a control lives and which page sequence reaches a goal; it is much weaker at saying how to compute an answer from a page already reached. Memory therefore appears to substitute for *exploration*, not *reasoning* (§6.3).

5.10 How much of the gain is left on the table

The oracle retriever—D's injection path and prompt template, but reading the raw pool and injecting the single trajectory recorded for the identical task rather than D's $k = 3$ compressed entries—reaches 53.4%, which is +12.0 pp above the full design and larger than the +11.4 pp we actually gained over baseline. It is an upper bound on retrieval, not a target: the injected entry is that task's own best run, which no first-step retriever could find, so part of the gap prices knowing the answer rather than searching better. All it supports is that retrieval is the stage a better method would have to improve; it says nothing about compression headroom, which the one-dimensional sweeps of §6.3 bound only at one run's resolution and the retrieval-size sweep locates in ranking rather than recall.

6 Discussion

The practical lesson is not "use memory" but "curate, compress, then retrieve": drop selection and the system lands below the no-memory baseline, drop compression or admission and it barely clears it. Keeping part of the design and not the rest buys very little.

The task-type gradient points the same way (§5): the effect amortises *exploration* rather than transferring *knowledge*; template reuse is not separated from it. Our retrieval is one cosine lookup on a frozen embedding, which is the part of the design we would expect to be most improvable.

6.1 Calibration against published numbers

Table 12 (Appendix C) places our two headline numbers against what has been reported on this benchmark; the comparison does not come out in our favour, and §C.9 states that reading in full rather than leaving the reader to find it.

6.2 What a report of a shared-memory system should carry

A success rate cannot separate a design that retrieves well from one that injects everything: the two differ in what the prompt holds, not in whether memory was consulted. Each condition therefore reports the pool and retrieval diagnostics of §4 beside its score: a rate says that a condition failed,

and those say why. Cost belongs in the same table for the same reason: retrieved material competes with a task's own observations for the context window, so a condition that gains while spending that window on other agents' trajectories should be asked what it displaced.

6.3 Limitations and Future Work

We keep our scaffold unoptimised to isolate the memory effect, so our absolute numbers are not comparable to systems that tune observation formatting, action space or step budget, and none of ours approaches the leaders on this benchmark.

The length control is run at D's budget, 6.3k, not at B's 18.6k: a condition that fills B's budget with text unrelated to the task would separate the two, and we do not run one (§C).

Two limits bear on the compression claim. First, the sweeps behind it are coarse and one-dimensional: widths of 32 to 256 dimensions, β over two orders of magnitude, the duplicate threshold over 0.85 to 0.98 (Figure 5). Every swept setting, G at 256 dimensions included, lands inside the ± 3 pp resolution of a single condition, so we read the three axes as flat at that resolution rather than as measured levers. That says no single knob is the lever behind the +11.4 pp headline; it does not say a finer setting would leave the rate untouched. Second, each sweep moves one axis with the other two held at deployed values, so the panels are silent about interactions.

A third group of choices is held rather than swept. Retrieval size is varied (Table 11), but one-dimensionally and against a fixed schedule: entries are fetched once, at the first step, and never refreshed, so the sweep bounds what a larger k contributes to a single prompt, not when a larger pool would have to be queried. The injected payload is a whole trajectory, so a finer unit—a step, or the sub-trajectory around the step that mattered—is not tested; the weak computation-type gain is a fact we report rather than a diagnosis we can act on. Context overflow is resolved by first-in-first-out eviction and we never vary that policy. D does not overflow at all, so no eviction rule could move its rate, and whether a better policy would rescue a condition that does is untested.

The VAE is trained per seed on that seed's pool, which keeps seeds independent but means the representation improves as the pool grows—accepted and logged, not eliminated. There is no freeze-after-warm-up arm, so the random-projection control prices whether the index is trained at all rather than when its adaptation should stop. Our compression claim is further bounded by the absence of a text decoder (§3). Every memory baseline is our own re-implementation, and the Synapse port floors that method rather than testing it fairly (§5). We report one benchmark in full, and its six sites and three task types may be idiosyncratic. The second environment we do add (Appendix C.10) is a transfer check at 3 seeds, read for direction and not for size, so the phenomenon is still untested on a fully powered second benchmark—the most important thing we have not done. The benchmark's template structure is likewise not separated from the mechanism: its 812 tasks are 130 intent templates of 6.2 on average, retrieval excludes the same task but not the same template, and part of D's gain could be memorising a template rather than transferring experience. The per-type breakdown is not a fully crossed design: both partitions are marginal recomputations of the same runs rather than independent estimates (§4); and one main model plus one cross-model check is weaker than a broad model sweep; the preliminary run of Appendix C.1 fixes only the arm ordering.

7 Conclusion

We showed that the assumption that shared experience helps web agents is false in its unqualified form: injecting everything is much worse than injecting nothing, and a curated pool read without a selection rule is worse too. The first finding is that selection, not sharing, sets the sign: the same pool reaches 41.4% under the intent rule and 20.1% when only that rule is randomised. Admission and compression move the margin, not the sign, and the index need not be learned. The +11.4 pp gain repeats under a second model family and is bounded from above by a +12.0 pp oracle gap that marks retrieval as the larger term we leave unexplored. The second finding is that memory amortises *exploration* rather than transferring *knowledge*, which the task-type gradient is consistent with but does not isolate, since template structure is not separated from it. We have tried to make the result checkable, not merely believable: conditions are points in one axis space, seeds never share a pool, and the statistics use the unit of randomisation. The negative result is the part we expect to survive replication.

AI USE STATEMENT

In this work, we used generative AI tools for code development (implementing, debugging and running the experiment and analysis scripts) and for editing the manuscript to improve readability and structure. We have not used generative AI tools for the remaining categories of disclosure, and they are not applicable to this work.

We have reviewed all AI-assisted work: every reported number, table entry and citation was reviewed by the authors against the source it comes from, and the released code was checked against the protocol the paper describes. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI. Generative models are not authors.

ETHICS STATEMENT

The experiments use simulated web services hosted locally and contain no human subjects, no personal data and no deployment in a live environment. API-based agents do send task text and page observations to a third-party model provider, which is a data-handling consideration for any real deployment but not for these controlled runs, whose content originates entirely from the benchmark's synthetic sites.

REPRODUCIBILITY STATEMENT

What this paper contributes is a set of paired comparisons rather than an absolute score, and the material below is arranged to support those comparisons. Section 4 fixes the benchmark, the pinned release and its checksum, the task composition, the models and the decoding parameters, and the unit of randomisation; Appendix A.7 gives the implementation of the memory scaffold, the retrieval key, the bottleneck architecture and the compressor's training schedule. Every condition is a point in the same three-axis space (Section 4.6), so one row of Table 1 differs from another by configuration rather than by implementation. The appendix also carries the full per-seed counts (Table 6), the hyperparameter sweeps (Figure 5), the per-task cost accounting (Table 8), the retrieval-size sweep (Table 11), the calibration against published numbers (Appendix C.9) and the second environment (Appendix C.10). Each episode appends one record to a per-(condition, seed) ledger, and every result value printed here is defined in one block at the top of the source rather than at the point of use; run order within a seed is randomised. Because temperature 0 does not make a served model bit-deterministic, we do not claim exact reproduction: A+tok against A re-runs one configuration and measures the protocol's reproduction floor directly, at +1.7 pp.

The code that runs these experiments is released in anonymised form at `https://anonymous.4open.science/r/Webagent-experience-sharing-9586`.

ACKNOWLEDGMENTS

We thank the maintainers of the open-source software this work builds on (the WebArena and WebArena-Verified benchmarks, and the agent tooling they depend on) and the providers of the model APIs used for the experiments, whose work made this study possible.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S: an open agentic framework that uses computers like a human. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samual Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vázquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? In *Forty-first International Conference on*

Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR / OpenReview.net, 2024.

Amine El hattami, Megh Thakkar, Nicolas Chapados, and Christopher Pal. WebArena verified: Reliable evaluation for web agents. In *Workshop on Scaling Environments for Agents*, 2025.

Jakob N. Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1146–1155. PMLR, 2017.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, 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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Genglin Liu, Shijie Geng, Sha Li, Hejie Cui, Sarah Zhang, Xin Liu, and Tianyi Liu. WebCoach: Self-evolving web agents with cross-session memory guidance. *CoRR*, abs/2511.12997, 2025a. doi: 10.48550/ARXIV.2511.12997.

Jiayuan Liu, Tianqin Li, Shiyi Du, Xin Luo, Haoxuan Zeng, Emanuel Tewolde, Tai Sing Lee, Tonghan Wang, Carl Kingsford, and Vincent Conitzer. The memory curse: How expanded recall erodes cooperative intent in LLM agents. *CoRR*, abs/2605.08060, 2026. doi: 10.48550/ARXIV.2605.08060.

Yitao Liu, Chenglei Si, Karthik R. Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 14179–14198. Association for Computational Linguistics, 2025b. doi: 10.18653/V1/2025.ACL-LONG.694.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of LLM agent memory mechanisms. In *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 41622–41652. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.FINDINGS-ACL.2069.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. *CoRR*, abs/2509.25140, 2025. doi: 10.48550/ARXIV.2509.25140.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pp. 2:1–2:22. ACM, 2023. doi: 10.1145/3586183.3606763.

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Chenghao Qiu, Chunli Peng, Yufeng Yang, Kuan-Hao Huang, and Yi Zhou. When correct demonstrations hurt: Rethinking the role of exemplars in in-context learning. *CoRR*, abs/2605.26350, 2026. doi: 10.48550/ARXIV.2605.26350.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 3980–3990. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1410.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

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 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, Hao Cheng, Chi Wang, and Wangchunshu Zhou. Agent KB: leveraging cross-domain experience for agentic problem solving. *CoRR*, abs/2507.06229, 2025. doi: 10.48550/ARXIV.2507.06229.

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. *Trans. Mach. Learn. Res.*, 2024, 2024.

Haoyu Wang, Guangyuan Dong, He Liang, Zijing Zhang, Jiachen Luo, Chuang Liu, Chao Xue, and Hao Tang. MemGuard: Persisting verifier signals for LLM-agent memory governance. *CoRR*, abs/2608.21867, 2026. doi: 10.48550/ARXIV.2608.21867.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation framework. *CoRR*, abs/2308.08155, 2023. doi: 10.48550/ARXIV.2308.08155.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Qi Zhang, and Tao Gui. The rise and potential of large language model based agents: a survey. *Sci. China Inf. Sci.*, 68(2), 2025. doi: 10.1007/S11432-024-4222-0.

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), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 623–645. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.27.

Wujiang Xu, Yu Wang, Kai Mei, Kaiqu Liang, Zhenting Wang, Mingyu Jin, Han Zhang, Shi-Xiong Zhang, Wenyue Hua, Sambit Sahu, and Dimitris N. Metaxas. MemGym: a long-horizon memory environment for LLM agents. *CoRR*, abs/2605.20833, 2026. doi: 10.48550/ARXIV.2605.20833.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. AgentOccam: A simple yet strong baseline for LLM-based web agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61349–61385. PMLR / OpenReview.net, 2024a.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19724–19731. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29946.

Ruiwen Zhou, Yingxuan Yang, Muning Wen, Ying Wen, Wenhao Wang, Chunling Xi, Guoqiang Xu, Yong Yu, and Weinan Zhang. TRAD: enhancing LLM agents with step-wise thought retrieval and aligned decision. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pp. 3–13. ACM, 2024a. doi: 10.1145/3626772.3657788.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: learning to synergize memory and reasoning for efficient long-horizon agents. *CoRR*, abs/2506.15841, 2025. doi: 10.48550/ARXIV.2506.15841.

A Definitions and Conventions

This section fixes what each reported quantity is, so that the tables in Appendix C can be read without reconstructing them from the main text, and states how the two partitions of the task set are cut.

A.1 Rates, slices and their spreads

Every rate in this paper is a task-level success rate: the fraction of the 812 tasks the deterministic evaluator scores as solved, collapsed to one outcome per task per seed by the majority rule of §4.6. A condition is run 5 times, so the number we report is the mean over seeds of that fraction, and the figure beside it is the sample standard deviation of the 5 seed means rather than of the tasks. Table 6 prints the 5 counts each rate is built from, so the three columns can be recomputed rather than taken on trust: their mean is the count in the main table, that count over 812 is the rate, and their spread is the SD printed next to it.

Two partitions of the task set are used. The difficulty split is the benchmark's own labelling: 258 tasks are hard and the remaining 554 are not, and the two exhaust the set. The type split is navigation, retrieval and mutation, at 113, 325 and 374 tasks. Both partitions are exhaustive, which is what makes each slice's delta a marginal of one flip table and what ties the per-type and per-difficulty figures back to the headline (Appendix B).

A.2 Retrieval and step-level quantities

Recall@3 is the share of injections whose three retrieved entries include one that a human judge marked relevant. Relevant and Irrelevant are two judgements of the same injected set and therefore partition it; Duplicates and Misleading are properties of the pool rather than verdicts on an injection—an entry is a duplicate when it near-duplicates another entry, and misleading when the annotation pass judged it so on injection—which is why a curated pool still reports both, and why neither is a count of entries the curation rule turned away.

Two properties of what reaches a prompt are worth separating here. The candidate set never contains the querying task's own trajectory: the retriever drops every entry whose recorded task id is the task being solved, in every mode except the oracle, which is the one condition built to read them (the candidate filter in `retriever.py`). This matters because the three agents of a seed walk the same seed-locked order one after another, so the pool does hold earlier agents' runs of the task in hand—which is why the share of same-task injections is zero by construction rather than small, and why Recall@3 is a cross-task quantity. The run records a same-*template* rate alongside Recall@3, computed between an injected entry's intent template and the querying task's; a same-*task* rate is not recorded, because under that filter it could only ever be zero.

The second environment is scored per step against a ground-truth action sequence, so its Step Success Rate is the share of reference steps an episode reproduces rather than the share of tasks it solves. That is why those columns are kept out of every comparison with Table 1.

A.3 The two intervals, and the resolution we read against

Differences are reported with a paired bootstrap over tasks: the same tasks, the same seed, and the resampling applied to the per-task paired outcomes, which is what makes two intervals in this paper comparable with each other. Because tasks drawn from one intent template are not independent, we also report a cluster bootstrap that resamples templates while keeping the task-level mean as the estimand; it is wider by a factor whose square is the design effect, 1.7, which corresponds to an intra-template correlation of 0.1 over templates of 6.2 tasks on average. The resolution a single condition is read at is ± 3 pp, the normal approximation $1.96\sqrt{p(1-p)/n}$ at the no-memory rate—a resolution set by the task count rather than fitted—so a gap below that is reported as unresolved rather than as an effect.

A.4 The Holm family, in full

Table 2 is the family the correction is applied over, printed with the counts behind every entry. The family is the memory conditions against A; the two length controls (A+tok, Filler) sit outside it,

because they are manipulation checks whose intended reading is that they are indistinguishable from A—correcting a check that is meant to come out null would correct for a hypothesis the paper does not put forward. It would also change the correction for everything else, because the largest raw p-value in a family is what Holm's step-down is anchored to: adding those two rows raises the bound we quote from 0.005 to 0.70. The bound is the family's largest adjusted value and not a separate computation—No-admit's, at 5.4×10^{-3}—which is why every other condition clears it. For each contrast b counts the task-seed pairs the condition wins and A loses and c the reverse; the raw p is the two-sided exact McNemar tail on the $b + c$ discordant pairs, the multiple is Holm's $m - r + 1$ at that rank, and the last column is the running maximum of the product, so it is monotone in the raw one by construction.

Table 2: The Holm family in full: each memory condition against A over the 812×5 pooled task-seed pairs, ordered by raw p. No-admit sets the quoted bound at 5.4×10^{-3}, printed as 0.005.

Condition	b	c	Raw p	Multiple	Adjusted p
Oracle	1441	490	3.4×10^{-108}	13	4.4×10^{-107}
B	307	1013	2.7×10^{-88}	12	3.3×10^{-87}
CF-failed	414	992	5.5×10^{-55}	11	6.1×10^{-54}
C	465	936	7.7×10^{-37}	10	7.7×10^{-36}
D	1087	625	3.6×10^{-29}	9	3.3×10^{-28}
F	500	903	3.2×10^{-27}	8	2.5×10^{-26}
D′	975	676	1.9×10^{-13}	7	1.3×10^{-12}
Reflexion	933	679	2.7×10^{-10}	6	1.6×10^{-9}
Rand-proj	931	691	2.8×10^{-9}	5	1.4×10^{-8}
Synapse	901	716	4.6×10^{-6}	4	1.9×10^{-5}
Expel	887	718	2.7×10^{-5}	3	8.1×10^{-5}
E	861	716	2.9×10^{-4}	2	5.7×10^{-4}
No-admit	853	741	5.4×10^{-3}	1	5.4×10^{-3}

A.5 THE TASK SET AS ACHIEVED

Table 3 reports the cross-tabulation the task set actually achieves, emitted by `build_tasksets.py` rather than designed. It is uneven in the direction that matters—Wikipedia supplies sixteen tasks against GitLab's hundred and ninety-six—which is why the per-type breakdown in Figure 3 is read as descriptive, and why we report the difficulty split alongside it: the hard subset cuts across all six sites and is not exposed to the same confound (§4).

The clustering unit of §4 is fixed by a normalisation rule rather than by any field the benchmark ships: an intent is lowercased, every character outside `a`–`z`, `0`–`9` and space is folded to a space, and the first two words that remain name its template. The fold replaces rather than deletes, so punctuation ends a word instead of joining two, and digits are kept; on the 812 tasks it gives the 130 templates of 6.2 tasks the main text reports. WebArena-Verified ships an `intent_template_id` of its own, which partitions the same tasks more finely, and we do not use it: the groups it defines are not the ones the cluster bootstrap resamples, and a cluster interval read against a finer partition than the one it was built on would overstate its own resolution.

Table 3: Achieved task-type by site cross-tabulation of the full task set. The three dashes are cells the benchmark leaves empty, not cells we left unfilled; the column totals are the 113, 325 and 374 task counts the slice analysis is weighted by.

Site	Navigation	Retrieval	Mutation	Total
Shopping	45	81	66	192
Shopping admin	18	86	78	182
GitLab	16	53	127	196
Reddit	—	11	103	114
Map	22	90	—	112
Wikipedia	12	4	—	16
Total	113	325	374	812

A.6 What each run records

Each episode records: step count; prompt, completion and cached tokens with a per-stage breakdown; end-to-end and per-call API latency; context-overflow evictions and which entries were evicted; the retrieved entry ids, their similarities and sites; Recall@k under two explicit definitions (a structural one and a strict template-matched one); the cross-site and same-template injection rates; pool size, duplicate rate, misleading share and per-site/per-type composition after each episode; and the verified evaluator's own version and checksums.

A.7 Implementation details

Agents run 15 environment steps at temperature 0 with a fixed action-space prompt; the observation is the accessibility tree. The encoder is `all-MiniLM-L6-v2`, frozen. The VAE is a $384 \to 256 \to d$ network with ReLU activations and batch normalisation, trained per seed with Adam (lr $= 10^{-3}$), early stopping with patience 30, and $\beta = 10^{-3}$. Its backward pass is hand-written in NumPy and is verified against central finite differences across all parameters before any result is produced. Payload character budgets are 1500 for raw storage and are set from the latent width for compressed storage. The rendering itself is deterministic and calls no model, so the text a retrieved entry contributes is reproducible from the trajectory alone. Context overflow is handled by deterministic FIFO eviction of injected memories, oldest first, with the system prompt and the current observation never evicted. Evaluation uses WebArena-Verified's deterministic evaluators; the evaluator version and dataset checksums are stored with every result record so that a later re-score can be audited.

B Derivations

Several of the numbers in this paper are combinations of other printed numbers rather than separate measurements, and more than one table carries a quantity that a second table also carries. Each relation the text relies on is listed here, in the form a reader can recompute from the printed values alone: no run data is needed to check any of them, and none is a fitted result, because both sides are read off the same records.

1. **Rates from counts.** $336.2/812 = 41.4\%$, and the same division holds for every row of Table 6; the delta column is the difference of two printed rates, $-17.4 = 12.6 - 30.0$ and $+11.4 = 41.4 - 30.0$.
2. **The split D′ exists to make.** $+7.8 = +4.0 + +3.8$: the gap between D and raw storage under the same retrieval key separates into the part the payload form accounts for and the part the index does.
3. **Slice weighting.** $(113 + 17.9 + 325 + 13.2 + 374 + 7.8)/812 = +11.4$ and $(258 + 13.8 + 554 + 10.3)/812 = +11.4$. Both partitions are exhaustive, so a size-weighted average of their deltas returns the headline rather than something near it.
4. **One measurement in two tables.** $+13.8 = 39.0 - 25.2$, and the same value is $+13.8$: the hard slice of Figure 3 and the V4-Flash row of Table 4 are the same 258 tasks in the same runs. The two rows of that table therefore differ in the model and in nothing else.
5. **The cluster interval.** The design effect is the squared ratio of the two interval widths and the intra-template correlation follows from it and the mean template size, both from numbers the main text prints.
6. **Subset intervals widen.** The cross-model interval covers 258 of 812 tasks, so it is wider than the interval it is a subset of by roughly $\sqrt{812/258}$; a subset interval *narrower* than the full-set one would mean the two were not the same kind of object.
7. **The ablation grid's summaries.** 3.9, 1.8 and 1.6 are best minus second best within a column; 4.9, 6.5 and 5.1 are the spread across keys within a row; 7.8, 4.7 and 4.7 the spread across representations within a column; 1.2 the largest residual of an additive fit to all nine cells. The grid's best cell is the deployed configuration itself, so 41.4 and 41.4 are one number: no cell of the grid beats the condition the tables report.
8. **The retrieval judgements partition.** 21 and 79 sum to 100, and so do the corresponding pairs for C, D and E.

Table 4: Cross-model check: success (%) on the 258-task hard subset, the same tasks for both rows. Only the GPT-5-mini row is a new run; the V4-Flash row is the hard-subset slice of the main-layer runs. Both intervals are the same paired bootstrap over tasks with the same seed, so the rows are comparable interval to interval rather than only point to point.

Model	A (258 Hard)	D (258 Hard)	D−A	95% CI (paired bootstrap)
DeepSeek V4-Flash	25.2 ± 2.4	39.0 ± 3.1	+13.8	[+10.3, +17.4]
GPT-5-mini	21.7 ± 2.1	32.4 ± 1.5	+10.7	[+7.4, +14.0]

Table 5: Retrieval quality and pool composition at steady state. The two judgement rows are over the same *injected* set, so they partition it and sum to 100 in every column—checkable, not stipulated. The two pool rows describe the *pool* rather than what entered the prompt: a duplicate near-duplicates an entry the pool already holds, a misleading entry is one the annotation pass so judged on injection. Neither is a rejection count, so their denominator is not the judgement rows'. B has no retrieval stage: Recall@3 is undefined for it and its relevance figures are rates among the entries it injects wholesale. No-admit is D with the admission rule switched off, so its two pool rows are what that rule was holding back. Rand-proj keeps D's rule and D's width, and its pool rows sit between D's and E's anyway: the duplicate test runs in the same space as the retrieval key, so an untrained projection separates entries less sharply than a trained one and more sharply than storing the raw vectors does.

Metric	B	C	D	E	No-admit	Rand-proj
Recall@3	—	0.13	0.64	0.47	0.58	0.51
Relevant (%)	21	33	78	66	71	68
Irrelevant (%)	79	67	22	34	29	32
Duplicates (%)	38	30	9	24	31	22
Misleading (%)	41	28	7	14	24	13

9. **The transfer table.** 1341 is 252 plus 177 plus 912; within a split the delta is the difference of the two rates and the interval is the paired bootstrap over that split's tasks, and the ordering D above A above B holds in all three.

10. **The sweeps.** Each panel's deployed setting is D's own rate, 41.4; the width panel carries a second ring at an already-run condition, 39.5; and 1.9 is the largest deviation from 41.4 over all 12 settings, which is what puts every one of them inside ± 3 pp.

C Additional Experiments

This section holds the runs whose result the wording of the main claim does not depend on, together with the two tables moved out of the main text for space. The quantities they report are defined in Appendix A. They are reported so that the mechanism can be checked rather than taken on trust: the preliminary run on a third model (§C.1), the retrieval and pool diagnostics, the per-seed counts behind Table 1, the storage-by-retrieval-key grid, per-task cost, the pool sweep, the retrieval-size sweep, the misleading-entry curve, the hyperparameter sweeps, and the calibration against published numbers. Table 4 is the cross-model row the main text discusses, repeated here with its interval. What these tables do and do not separate is taken up in Appendix D.

C.1 A preliminary run on a third model

The design the rest of this paper reports was settled by a smaller run made first, on Qwen3.5-9B rather than DeepSeek V4-Flash, and it is reported here because it is the only measurement in the paper that comes from outside the design: it ran 12 tasks drawn from four sites, 3 agent seeds per task and a cap of 15 steps per episode, so each condition it compares is 36 agent-task runs. Three of its conditions are the A, B and D of this paper. No memory solved 4 of those runs (11.1%); sharing every trajectory with no filtering solved 0 (0.0%); and compressing each trajectory into a 128-dimensional latent retrieved by intent similarity solved 6 (16.7%). That is the ordering the main layer reports, and the 7 further conditions it ran—random selection, raw trajectories behind the same similarity key, the wider latent, and combinations of those—all landed at or below the no-memory arm, the best of them at 8.3%.

The run cannot support anything beyond that ordering. The two arms that matter differ by two solved runs, and at that size no exact test separates them: $p = 0.500$ for D against A and $p = 0.125$ for A against B, over the paired agent-task cells. Its rates are also far below the main table's, which we attribute to the smaller model, to the size of its sample, and to a scaffold that was still being changed while it ran. We report it as the reason the main layer was built the way it was, not as evidence for what the main layer finds. What makes it worth printing anyway is the ablation pattern: the claim that compression with similarity retrieval is what pays, and that random selection and a wider latent do not, reads off this run without needing a larger sample, and that is a weaker claim than the tables make, but it is the strongest one this record supports.

C.2 Per-seed counts

The counts every rate in Table 1 is built from, printed so that the mean, the rate and the SD can be recomputed from the paper alone rather than taken on trust. The identity that ties the 5 columns to the printed rate is the first item of Appendix B.

Table 6: Tasks solved per seed, one row per condition; the column labels s0–s4 abbreviate seeds 0–4. The *Solved* and SD columns are Table 1's, not a second computation: the mean of the five counts is *Solved*, their sample SD over 812 in points is the SD, and the mean over 812 is the rate. That is how Table 1 is meant to be checked.

Condition	s0	s1	s2	s3	s4	*Solved*	SD
A	243	260	228	253	235	243.8	1.6
A+tok	276	238	260	261	253	257.6	1.7
Filler	244	259	243	236	221	240.6	1.7
B	100	105	85	119	104	102.6	1.5
C	167	146	131	152	152	149.6	1.6
D	332	347	323	353	326	336.2	1.6
No-admit	257	275	267	250	282	266.2	1.6
D′	308	285	323	302	300	303.6	1.7
E	269	254	272	292	277	272.8	1.7
Rand-proj	301	283	276	308	291	291.8	1.6
F	179	172	147	154	164	163.2	1.6
CF-failed	137	129	112	119	144	128.2	1.6
Oracle	413	434	428	455	440	434.0	1.9
Reflexion	299	293	314	276	291	294.6	1.7
Synapse	300	277	285	280	262	280.8	1.7
Expel	274	297	276	282	259	277.6	1.7
G	341	316	319	301	326	320.6	1.8

C.3 Storage and retrieval key

The grid varies the two axes the method has—what is stored and how it is selected—with the pool and the prompt template held fixed within a row. The three keys select on the task intent alone, on the first step's accessibility tree alone, or on the two concatenated. Because the query is encoded rather than injected, the choice among them carries no prompt cost, which is why the grid's columns are comparable without a budget correction. Both the summaries and the reading they support are in the sections named above.

Table 7: Storage $\times$ retrieval-key ablation, success (%) on all 812 tasks. Each cell is a full-set rate. The column axis is clean except in the raw row, whose intent cell is E—payload, pool and prompt template are fixed within the other rows—the row axis is not (§5.8).

Storage \ Retrieval key	intent	page state	mixed
raw trajectory	33.6	30.2	35.1
VAE-128	41.4	34.9	39.8
key-point summary	37.5	33.1	38.2

C.4 PER-TASK COST

Cost is measured per task rather than per solved task, so the columns are comparable across conditions and are not flattered by the ones that solve more.

Table 8: Per-task cost on the main layer. D's prompt is about $1/3$ of B's while its step count is 27% below A's. Refl. is our Reflexion re-implementation, and its cheaper prompt reflects a shorter carried context, not a cheaper method, so it is not a like-for-like row.

Per-task cost	A	B	D	E	Refl.
Steps	13.2	15.0	9.6	11.8	12.4
Prompt tok. (k)	4.1	18.6	6.3	9.8	7.1
End-to-end (s)	128	214	102	145	156
API (s/step)	3.1	6.8	4.2	5.1	4.5
Context overflows	—	47	0	3	0

The table above measures four kinds of cost, which is what a reader needs to judge the method; it does not cover every condition, because steps and latency are not read for the conditions that exist to isolate one mechanism. Prompt tokens are, since they are the x axis of Figure 2, so they are given for all 16 conditions in Table 9 below. The four that the table above also carries are the same numbers; nothing here is a second measurement.

Table 9: Mean prompt tokens per task for every condition of Table 1, in thousands, which is the quantity Figure 2 plots on its x axis. G is reported without a table row and is not plotted there, so it has no entry.

Condition	Prompt tok. (k)	Condition	Prompt tok. (k)
A	4.1	CF-failed	4.6
A+tok	4.1	Oracle	11.2
B	18.6	Reflexion	7.1
C	6.2	Synapse	10.4
D	6.3	Expel	8.4
D′	7.2	Filler	6.3
E	9.8	No-admit	6.4
F	4.4	Rand-proj	6.3

C.5 POOL SIZE

The sweep in Table 10 is run on D's own configuration, so its deployed column is the D column of Table 5 and the two cannot drift apart; the reading is in Appendix D.

Table 10: Retrieval quality and latency against pool size, condition D. Five points, no replicates, no fitted breakpoint: the curve is reported, not modelled.

Pool size (entries)	10	50	100	200	400
Recall@3	0.81	0.72	0.64	0.58	0.52
Relevant (%)	86	81	78	74	71
Duplicates (%)	4	7	9	12	15
Retrieval latency (ms)	18	24	31	42	58

C.6 RETRIEVAL SIZE

The sweep moves one axis, the number of entries injected, with the pool, the payload budget and the prompt template held. Its prompt-cost row is affine in k rather than free—the injected block is k payloads at a per-entry budget, and the query is encoded rather than injected—which is also why a richer retrieval key costs nothing in prompt tokens and the three keys of Table 7 are compared at equal cost. Neither property is what separates the keys; the cost of sweeping this axis is the entries it admits, not the query it searches with.

What the sweep buys is a reading of the oracle gap that §5 leaves qualitative. Recall rises monotonically across the range, from 0.64 at the deployed $k = 3$ to its highest value at $k = 12$, while the

Table 11: Condition D's retrieved-entry count swept over 6 settings from 1 to 12. The column at $k = 3$ is the deployed configuration, so all three of its cells are D's numbers from Tables 1, 5 and 8 rather than numbers of its own. One axis, no replicates: the shape is reported, not modelled.

Retrieved entries (k)	1	2	3	5	8	12
Success (%)	37.6	40.2	41.4	41.0	38.9	35.7
Recall@k	0.41	0.55	0.64	0.72	0.78	0.82
Prompt tok. (k)	4.8	5.6	6.3	7.8	10.0	12.9

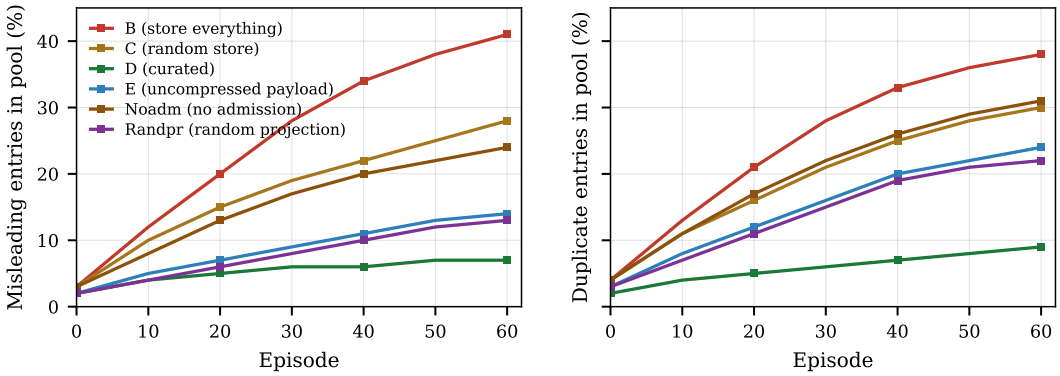

Figure 4: Two properties of the same pool against episode index, one line per condition and one panel per property: share of entries the annotation pass judged misleading (left), and share that near-duplicate another entry (right). Both describe the pool that survives curation rather than counting what was rejected, which is why D's lines are low rather than absent. Every line ends at that condition's cell in Table 5, so the last point and the table cell are one number read two ways. The two panels agree on the extremes, with D lowest, B highest and the removal arms between, but not further in: Noadm sits below C on misleading entries and above it on duplicates, which is the one order the two properties do not share.

success rate is flat across the interior and ends +5.7 pp below the deployed setting. The entries a larger k adds are found and are not used, which is the signature of a ranking limit rather than a recall one; and with the prompt at $k = 12$ still well inside B's, the fall is not overload either. What the oracle half of the gap prices is therefore neither search nor budget but the one thing no retriever can have—that task's own run.

The sweep also brackets B's budget from below: at $k = 12$ the prompt reaches 12.9k tokens—past D's 6.3k, inside B's 18.6k—and the rate is 35.7%. That is not a length control. Raising k adds entries and lengthens the prompt together, so the two move as one, and every entry at $k = 12$ is admitted, compressed and retrieved, which is what B lacks. It is not the control the negative result needs, and a condition that fills B's budget with text unrelated to the task is not run. The length control we do run, Filler, matches D's 6.3k and therefore bounds that argument only at its own length.

C.7 What the pool is made of, over time

C.8 Hyperparameter sweeps

One setting at a time around the deployed configuration, read for flatness at the resolution of a single run rather than for a preferred setting.

C.9 Calibration against published numbers

The aim of Table 12 is narrow: the no-memory baseline should sit below a tuned no-memory scaffold, so that it is not itself a competitive agent. That is checkable from the table and holds—A is 30.0% against AgentOccam's 43.1% on the same evaluator. It does not settle where the full design sits. The strongest anchor we could trace to a primary source sits above us rather than below it, and that is the honest reading of the table rather than a caveat on it. What the comparison supports is narrower than a claim over the leaders—it shows that memory buys what a better scaffold also buys, not that it

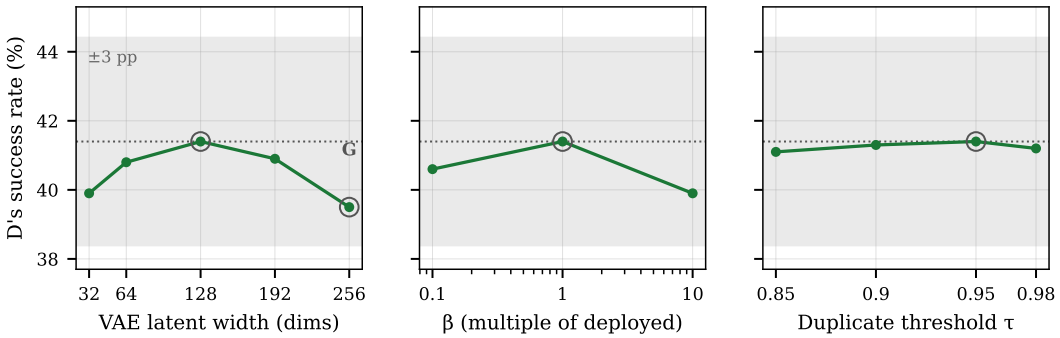

Figure 5: Condition D's success rate against the three hyperparameters it configures, one panel per sweep, 12 settings in all. Widths from 32 to 256 dimensions; β from 10^{-1} to 10^{1} times the deployed weight; duplicate thresholds τ from 0.85 to 0.98. The dotted line is 41.4%, D's rate in Table 1, and the grey band is ± 3 pp, the spread of a single condition across the five seeds. The ring marks each panel's deployed setting—the value D itself uses, which is why that point cannot differ from the row above—and the width panel carries a second ring at 256, which is condition G of Table 7. Every swept setting lands inside the band, the largest deviation being 1.9 pp at the widest bottleneck, so at the resolution one run has, none of the three moves the rate on its own.

buys more—and the systems above us are reported on a different task-set version and against their authors' own baselines, so the remaining gap is not attributable either. The anchors are on the original evaluator while our rows are on the Verified one, which makes the comparison indicative rather than controlled. Both published rows name their backbone because the backbone is part of the claim: AgentOccam's WebArena headline is on GPT-4-Turbo and AWM's on `gpt-4-0613`, and quoting either without it would attribute to a method what belongs to the model behind it.

Table 12: Calibration against published numbers. Each published rate is the one its authors improved on from their own scaffold—AgentOccam 37.2 to 43.1, AWM 23.5 to 35.5 over its baseline—so the gain a reader sees is the gain its author reports rather than one we recomputed. The published rows use the original WebArena evaluator and our rows the Verified one, and the relative-gain row is as its author reports it, so no published entry is directly comparable to our absolute rates: the table places our numbers, it does not rank them.

System (as reported)	Result
GPT-4 + CoT, no unachievable-task hint (Zhou et al., 2024b)	14.4%
AgentOccam (GPT-4-Turbo), original evaluator (Yang et al., 2025)	43.1%
AWM (GPT-4, `gpt-4-0613`), gain from memory (Wang et al., 2025)	+51.1% rel.
A (ours, no memory), 812 / Verified	30.0%
D (ours, curate + compress + retrieve)	41.4%

C.10 A SECOND ENVIRONMENT

The three splits in Table 13 are replayed offline against cached snapshots rather than browsed live, so this is a migration check rather than a second interactive benchmark. What carries over and what is rebuilt is worth stating exactly, because the two differ. Carried over unchanged: the scaffold,

Table 13: The second environment: 1341 tasks over the three published test splits of Deng et al. (2023), 3 seeds, Step Success Rate (%) and its per-seed spread. A, B and D are the three conditions Table 1 compares, each row paired against A on the same tasks, and the interval is the paired bootstrap over tasks used throughout, at the same seed. The published column is the benchmark's own figure under a different model, so our rates are expected to be lower and are not comparable to it. Step SR is scored per step against a ground-truth action sequence, so these columns are not on the same scale as Table 1, which counts tasks.

Split (n)	A	B	D	Δ (D−A)	95% CI	Published
Cross-Task (252)	24.6±1.4	17.9±1.6	32.5±1.8	+7.9	[+4.0, +12.0]	36.2
Cross-Website (177)	20.3±1.5	14.7±1.7	27.5±1.4	+7.2	[+3.0, +11.1]	30.1
Cross-Domain (912)	11.2±1.1	8.3±1.2	15.4±0.9	+4.2	[+2.4, +5.9]	26.4

the prompt template, the frozen sentence encoder, the retrieval key, the admission and duplicate thresholds, the payload character budget and the retrieval size k. Rebuilt for this layer: the VAE, retrained per seed on that seed's own Mind2Web pool, matching the per-seed rule of the main layer. What is asked of the layer is whether the direction survives a change of environment. It does. D beats the no-memory baseline on all three splits, and the store-everything counterexample falls below it on all three (-6.7, -5.6 and -2.9 points), so the ordering matches Table 1. The magnitudes do not: +7.9, +7.2 and +4.2 points are each smaller than the +11.4-point result on WebArena, and although all three intervals exclude zero, on the two smaller splits they are wide enough that we would not put a size on the transfer beyond its sign. It is built at 3 seeds against the main layer's 5, on splits from 177 to 912 tasks, so its spreads are coarser than anything else here; what it is evidence for is that the mechanism reproduces, not that the effect size does.

D Additional Discussion

Reading the diagnostics. D's Recall@3 is 0.64 against 0.47 for E and 0.13 for C, and its human-judged relevance rate is 78% against 21% for full sharing. Mediation attributes roughly 4–5 pp of success-rate change per 0.1 of Recall@3 under the indirect-effect decomposition, with a bootstrap indirect effect excluding zero. Recall is not randomised, so we make no causal claim from that slope; what supports the mechanism is the counterfactual controls in §5, which move the retrieval stage alone. The reading the slope is consistent with is the narrow one: curation decides what is in the pool, retrieval decides what reaches the prompt, and it is the second step that a compressed payload makes room for, since it governs how much of a fixed context another task's experience may occupy.

Where the pool curve stops telling us anything. Sweeping the pool by size, Recall@3 and relevance fall monotonically while duplicate share and retrieval latency grow, with no point at which quality turns up again. Where that decay becomes steep enough to matter is a judgement read off the curve rather than a quantity we measured: the marginal loss per added entry is largest below roughly 100–200 entries, but with five points and no replicates we fit no breakpoint. The practical claim is the weaker one—adding experience buys progressively less while costing more in latency—and it is also consistent with B degrading rather than plateauing, since an uncurated pool moves along the same curve in the worse direction at every episode.

What would have argued against the account. Four observations would have. If injecting *any* memory had matched our condition, the effect would be about pool membership rather than about what is retrieved, and the two controls that inject the wrong memories are the test: they land at -9.9 and -14.2 pp against A, below the no-memory baseline rather than at our level. If curation had left the pool's composition alone, the duplicate and misleading rows of Table 5 would not fall from B through C to D; they fall 38 to 30 to 9 and 41 to 28 to 7. If the direction were an artefact of one model's prompting, the second row of Table 4 would have reversed it. If it were a property of one environment's evaluator, the ordering on the second benchmark would have differed, and D leads A and B on all three of its splits (Table 13). Each is a single contrast, and none of them establishes generality; what they do is close the four cheapest alternative accounts one at a time.

Why the admission rule excludes failures. ReasoningBank distils from failed experiences as well as successful ones and reports that this beats storing successes alone, so the success-only rule needs an answer rather than an assumption. Two things separate the settings. It distils failures into strategies instead of putting a failed trajectory into the prompt, and the counterfactual prices the latter: injecting failed trajectories in place of successful ones lands at -14.2 pp against A, below the no-memory baseline. And its memory is written and read by the same agent, where a failure is corrective about one's own behaviour; ours is read by a different agent on a different task, where a trajectory that failed says little about what to do instead. Whether distilled failures would earn a place in a shared pool is a question this grid does not test: the admission rule governs what enters raw, and is not a claim that failure carries no information. The route such a pool would most plausibly take is the one MemGuard (Wang et al., 2026) sets out, where a verifier's signal is written onto each record as persistent metadata and goes on to govern admission, weighting and conflict resolution rather than being spent on a single accept-or- reject decision at write time. Our rule is the degenerate version of that, with the verifier removed and one decision taken instead of an ongoing one. We do not test the

richer form; its reported gains are on web-agent benchmarks close enough to ours that the comparison is worth a run, and we leave it to one.

What the grid does not separate. D, D′ and E read the same pool and use the same retrieval key, and move the payload form and the storage representation one at a time, which is what licenses reading +4.0 pp of the +7.8-point gap as the payload's and +3.8 pp as the index's—except that E carries the success-only rule where D and D′ carry success-plus-deduplication, so the index's share is at most +3.8 pp: E's rate is held down by the duplicates it admits, and part of that +3.8 may belong to the deduplication step rather than to compression. Curation and representation are varied together everywhere else, since the uncurated condition stores raw trajectories in full, so the grid cannot say whether the two interact; and the ablation's row axis changes pool composition along with the encoder that produces the entries, so a difference between its rows is not attributable to the representation alone (§5.8). Crossing curation with representation at a fixed payload is run at one point rather than not at all: No-admit holds the representation, key, payload and k at D's values and switches off only the admission rule, which is what makes its +8.6-point cost attributable to curation rather than to the payload that moves with curation elsewhere on the grid (§5). The rest of that crossing—curation off with raw storage at the same payload—is still the experiment we did not run, so we report the two as separate contributions rather than as an interaction.

What B's prompt is made of. A and B are given the same task, so the difference between B's prompt and A's is other agents' experience: 78% of B's prompt is retrieved material, which is the share the crowding-out reading turns on. The claim the paper makes from it is the narrow one—at B's setting the retrieved trajectories take the larger half of a fixed window, so what the agent needs to finish the task it is on is competing with what other agents did on tasks it is not—and not that any injection at any share would have the same effect, which nothing here varies.