CHOOSING THE TASK ORDER RETURNS MORE THAN GOVERNING WHAT GETS WRITTEN

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving LLM agents improve by writing their own experience into a memory bank and reusing it on later tasks. Prior work has focused on *what* the bank should keep: which experiences to admit, distil or prune. We show that a variable one step earlier can matter more: the *order* in which the same tasks arrive. On execution-graded code benchmarks, permuting a training stream while holding its content fixed moves held-out success by 4–11 points, and the spread shrinks to evaluation noise once memory is disabled, locating the effect in what the agent writes. Order changes which experiences exist, and filtering them afterwards does not recover the gain: at identical cost per deployed run, choosing the order adds 1.71 points on average, while write-time governance and seven further admission and retrieval rules yield no significant gain. We turn this finding into *the ordering lever*, a pilot protocol that briefly tries each candidate order on the target task pool and commits to the winner. It raises all four write policies on HumanEval+ by 2.3–3.2 points, including Reflexion, ExpeL and Agent Workflow Memory, recovers over 90% of the best candidate's gain on 34 freshly drawn task pools, and comes with a reliability score, computed from the pilot alone, that tracks its payoff. The best order is specific to each writer: installing one writer's winning order on another usually hurts, while that writer's own pilot helps. For agents that learn from their own experience, arrival order is a design choice to be measured, not copied.

1 INTRODUCTION

Self-evolving agents improve by writing experience into a persistent memory bank and retrieving it on later tasks (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024b; Ouyang et al., 2026; Park et al., 2023; Xu et al., 2026; Chhikara et al., 2025). Think of an apprentice who keeps a notebook while working: what ends up in the notebook shapes how the next job goes. This literature is almost entirely about what belongs in the notebook—which trajectories deserve an entry, how to distil and retrieve them, when to prune—and rarely about which job the apprentice is handed first. The evaluation template matches it: one canonical stream, a single held-out number, arrival order averaged away as nuisance.

Ye et al. (2026) showed that it is not: the default order of prior work is an implicit curriculum, and shuffling it turns a reported $+1.5$ point improvement into a -4.5 point drop. They diagnose the fragility and prescribe stricter reporting. What they leave open decides whether order is a hazard or an instrument: *where* does the effect live, and *can a deployer act on it*? We answer both: the bank carries the effect (§3), and a short pilot on the deployment pool captures it (§5). Figure 1 names the two places a designer can intervene; we call the instrument at the upstream one **the ordering lever**.

Order is a first-class variable, and the bank is what carries it. Permuting a training stream into twelve orderings while holding task content fixed moves held-out, execution-graded success by 4 to 11 points, significant in 11 of 12 decision units under a Holm-corrected permutation test. That range is the exposure of a single published number; what a deployer collects is the held-out gain of §5, about two tasks per pool. An effect that bypassed memory—drift in decoding or grading over a long run—would survive disabling the bank. It does not: with the bank disabled the same orderings spread by $1.6/1.3/2.2$ points—one to three test cases, the evaluator's own grain—and an order chosen for the memory-equipped agent transfers nothing to the memory-free one. And the same order can help one writer and hurt another: order is a sequence-by-writer interaction.

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

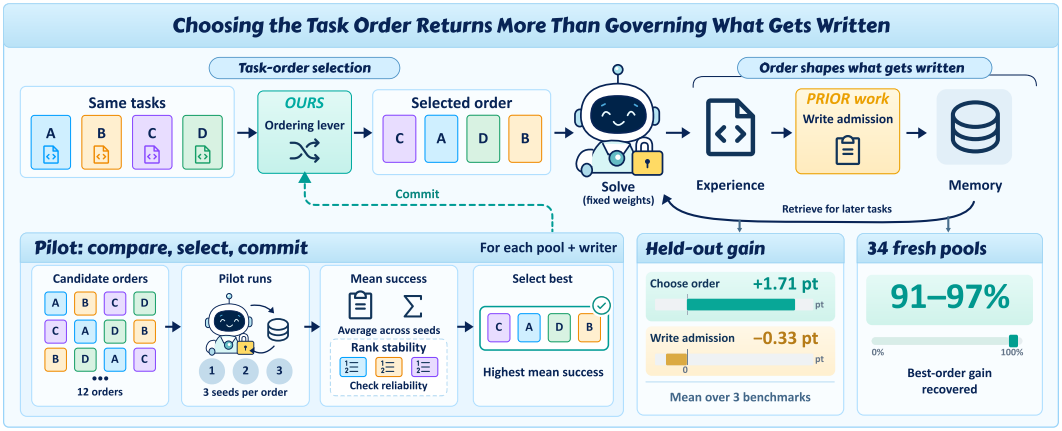


Figure 1: **Two places to intervene in a self-evolving loop.** Prior work governs write admission, deciding what the bank may keep; we act upstream, choosing the order in which the same tasks arrive. A short pilot compares candidate orders and commits the best one.

Order selection returns more than governance at equal per-run cost. We test payoff with a 2×2 over write governance and order selection on the same arms, seeds and estimator, at identical cost per deployed run. Averaged over MBPP, HumanEval+ and LiveCodeBench, order selection is worth $+1.71$ points and our governance -0.33, a difference whose 95% interval is $[+1.43, +2.61]$; order leads by 3.7 and 2.5 points on HumanEval+ and LiveCodeBench and ties on MBPP ($+0.88$ against $+0.94$). The pilot's one-off search is priced separately (§5). Seven further interventions, including a greedy admission rule given each entry's estimated effect, show no significant gain.

The best order must be measured, not copied. Because order interacts with the writer, a winning order does not travel: copying our pilot's order onto a baseline writer is significantly harmful in seven of nine cells, while that writer's own pilot helps in six. The deliverable is therefore a protocol: for each candidate order, run a short pilot on the deployment pool, commit to the winner, score on unseen seeds. This raises our gated writer and in-harness replicas of Reflexion, ExpeL and AWM by 2.3–3.2 points on HumanEval+, recovers 91–97% of the best candidate order's gain on 34 freshly drawn pools across three benchmarks, and comes with a reliability statistic that tracks, pool by pool, how much the pick will pay.

Contributions.

1. **Measurement.** A permutation-controlled study of arrival order in self-evolving agents: 4–11 points of held-out exposure on fixed content, carried by the memory bank, dependent on the writer, and stable across a doubled pool, ten seeds and 7B–72B backbones.
2. **Comparison.** A like-for-like 2×2 at identical per-run cost in which choosing the order adds $+1.71$ points while the write governance we test adds none, with seven further admission and retrieval rules also short of a significant gain.
3. **Method.** The ordering lever, a pilot protocol that re-measures the order for each writer (a copied order hurts in seven of nine cells), raises all four write policies on HumanEval+ ($+2.3$ to $+3.2$ points), recovers 91–97% of the best-order gain on 34 freshly drawn pools, and carries a pilot-only reliability score that tracks its payoff.

2 Measuring Order Effects

2.1 Setting

We study a fixed workload executed repeatedly with score feedback: an operator owns a pool of tasks, runs a self-evolving agent over it more than once, and can grade the outcome. This is the regime in which an operator can choose the arrival order, and the regime our deployment claims are about; transfer of the procedure to new pools is tested on 34 freshly drawn pools (§5).

Table 1: **The order effect, in-sample and held-out.** Governed writer. *Spread*: best−worst of twelve orderings, mean (range) over three streams. *Held-out*: orders picked on seeds 0–4, scored on 5–9, 95% interval; *in tasks*: divided by one test case. *Descriptive (§2).

	pool geometry		effect of reordering the same tasks		
benchmark	train / test	1 test case	spread (pt)	held-out best−worst	in tasks
MBPP	99/141	0.71 pt	7.7 (7.3–8.3)	+2.98 [+2.55, +3.40]	4.2
HumanEval+	85/75	1.33 pt	8.3 (6.4–10.7)	+5.60 [+3.73, +7.20]	4.2
LiveCodeBench	110/130	0.77 pt	4.9 (4.1–5.5)	+2.00 [+0.46, +3.69]	2.6
Mercury*	90/90	1.11 pt	8.3 (5.3–10.3)	+6.27 [+4.71, +7.82]	5.7

2.2 Agent, benchmarks and orderings

The agent is a retrieve-then-generate coder over a flat textual memory bank: after each training task it may write an entry, and before each later task it retrieves a short context. Richer scaffolds (Yao et al., 2022; 2023; Wang et al., 2022; Zhou et al., 2022; Madaan et al., 2023; Schick et al., 2023) are orthogonal to the stream. Confirmatory runs use `Qwen2.5-14B-Instruct` (Yang et al., 2025) at temperature 0, with a 7B–72B sweep below. We evaluate on MBPP (Austin et al., 2021), HumanEval+ (Chen et al., 2021; Liu et al., 2023), LiveCodeBench (Jain et al., 2025) and Mercury (Du et al., 2024), with train/held-out draws frozen in a manifest (99/141, 85/75, 110/130). Success is binary execution against official unit tests, so one held-out test case equals 0.71, 1.33 and 0.77 points—the unit in which we report every effect size. Each stream is permuted into twelve orderings of identical content: eight seeded random permutations and four constructed ones (easy- and hard-first by reference-solution length, blocked-by-family, interleaved). The first two assume an offline pool with reference solutions; §5 also reports the pilot restricted to orders that need none.

2.3 Write policies and pre-registration

All writers share one store that records successful trajectories, merges them into skill entries and caps each memory level at 900 tokens. The ungoverned control admits every such trajectory; our writer adds a $k{=}2$ evidence gate with provenance and periodic reorganisation; Reflexion (Shinn et al., 2023), ExpeL (Zhao et al., 2024) and AWM (Wang et al., 2024b) are reimplemented in the same harness; a no-memory control disables writes and retrieval entirely. At temperature 0 a seed reruns an identical configuration, so seed-to-seed variation is the nondeterminism of batched GPU inference; the no-memory arm measures that floor directly. Seeds 0–2 fit, 3–4 confirm during development, and 5–9 are reserved for a single confirmatory judgement, every contrast on them pre-registered with falsification thresholds before unblinding. Mercury was pre-registered as descriptive before unblinding, because its development-seed rankings were unstable (Pearson -0.23 between seeds 0–1 and 2–4); its rows carry an asterisk. Unless noted, intervals are seed-clustered bootstrap 95% intervals (Efron, 1992) over 10,000 replicates with a Holm correction across families (Holm, 1979), following work showing reported gains are routinely smaller than the variance they are measured against (Henderson et al., 2018; Dodge et al., 2019; Bouthillier et al., 2021; Agarwal et al., 2021; Pineau et al., 2021).

3 The order effect and its carrier

3.1 Size of the effect

We first measure how far reordering fixed content moves held-out success. Write $y(o, s)$ for held-out success under ordering $o \in \mathcal{O}$ and seed s, and define the order spread on a decision unit as

$$\Delta \;=\; \max_{o\in\mathcal{O}} \overline{y}(o) \;-\; \min_{o\in\mathcal{O}} \overline{y}(o), \qquad \overline{y}(o) = \tfrac{1}{|S|}\textstyle\sum_{s\in S} y(o,s). \tag{1}$$

Across twelve decision units (four benchmarks × three independently sampled streams), Δ ranges from 4 to 11 points and is significant in 11 of 12 under a Holm-corrected permutation test (Appendix C opens every unit). Figure 2(a) shows one unit—twelve runs over identical content, separated by eleven held-out test cases—and (b) shows every unit far above the evaluator's grain. The

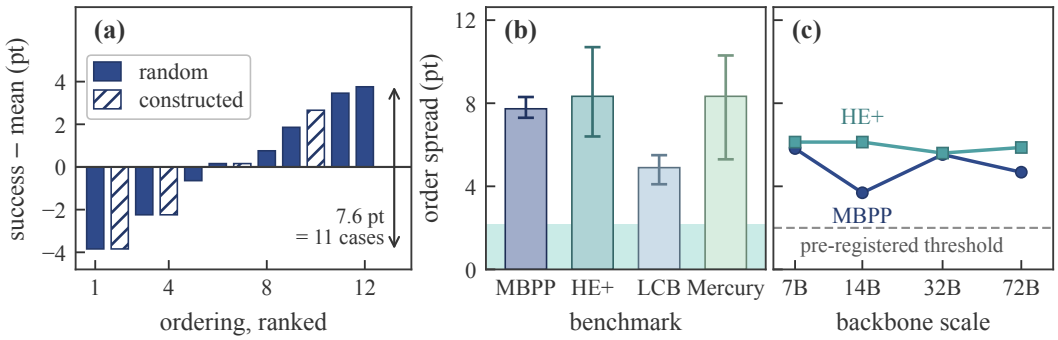


Figure 2: **Same tasks, different order, different agent.** (a) MBPP, stream s_0: twelve orderings ranked, as deviation from their mean. (b) Spread (Eq. 1) per benchmark: mean of three streams, whiskers from minimum to maximum; band: about three test cases. (c) Spread across backbones against the pre-registered threshold. HE+: HumanEval+; LCB: LiveCodeBench.

gap survives the winner's curse (Table 1): picking the best and worst orders on seeds 0–4 and scoring them on unseen seeds 5–9 still separates them by $+2.00$ to $+5.60$ points on the three confirmatory benchmarks, 2.6 to 4.2 held-out test cases, every interval excluding zero.

Robustness. Extending the main pool to ten seeds leaves single-run spreads in the same range ($p \leq 0.001$ on every benchmark), and the held-out reading of Table 1 is significant on all four. Nor is the spread a small-model pathology: sweeping the backbone over 7B, 14B, 32B and 72B on the ungoverned arm, MBPP reads $5.82/3.69/5.53/4.68$ points and HumanEval+ $6.13/6.13/5.60/5.87$ (Figure 2c, Table 9), all eight cells clearing the pre-registered 2-point threshold with no downward trend at the largest backbone, and doubling the MBPP pool from 240 to 480 problems raises the spread to 5.19 rather than diluting it.

What is stable is the gap, not the winner: the best order changes from unit to unit, so easy-first is not a free curriculum. An unreported order puts six to fifteen held-out tasks at stake; a deployer who chooses the order gains about two per pool (§5).

3.2 The memory bank as carrier

A frozen model can carry one task's lesson to the next only through its bank, so disabling the bank asks whether anything else—drift in decoding, grading or scheduling over a run—carries order. Two controls show nothing does. Disabling writes and retrieval collapses the same twelve orderings to a spread of $1.6/1.3/2.2$ points on MBPP/HumanEval+/LiveCodeBench—one to three test cases, indistinguishable from evaluator grain (Figure 3a)—and the ordering selected for the memory-equipped agent has no detectable effect on the memory-free one (MBPP -0.11 $[-0.44, +0.22]$; HumanEval+ $+0.07$ $[-0.58, +0.69]$; LiveCodeBench $+0.18$ $[-0.63, +1.18]$). The complementary control keeps the bank but removes governance, and the sign reverses: on HumanEval+ the interleaved order worth $+2.87$ $[+2.02, +3.56]$ under our writer is worth -2.33 $[-2.58, -2.04]$ ungoverned (Figure 3b). The ordering holds on MBPP ($+1.50$ against -0.59) and LiveCodeBench ($+0.33$ against -0.68). An ordering is neither good nor bad in itself; it is good *for a writer*, and this sign flip is the root of the transfer boundary of §5.

If the bank carries the effect, the natural target is what enters it; §4 tests that target and returns to which step of the pathway order acts on.

4 The reach of write-time governance

4.1 Governance versus order selection

Section 3 shows that order acts through the bank. A lever that acts earlier need not pay more, so we compare choosing the order with governing admission in one design. Table 2 crosses write

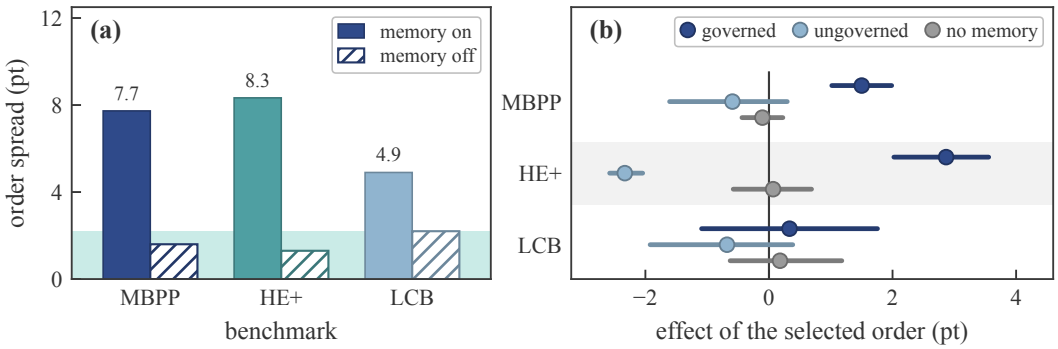


Figure 3: **The effect lives in the bank.** (a) Order spread with memory on and off, mean of three streams; band: about three test cases. (b) Each benchmark's selected order under three banks: dot, estimate; line, 95% interval.

governance with order selection on the same arms and seeds, every *selection on* cell running the pilot on its own arm. Each deployed run costs the same in all four cells (Table 7); the pilot's search is a separate, one-off cost that §5 prices. Averaged over the three pools, order selection returns $+1.71$ points and our governance -0.33, a difference of $+2.04$ $[+1.43, +2.61]$; on MBPP the two are level $(+0.88, +0.94)$, on the other two order selection leads by 3.7 and 2.5 points, both intervals excluding zero. The gate is not idle while earning that: it discards five sixths of the entries and seven eighths of the tokens the ungoverned bank keeps, yet moves MBPP success by under a third of a point. Removing most of what the agent wrote changes almost nothing.

Table 2: **Ablation: which lever carries the return.** Held-out success, seeds 5–9, equal cost per deployed run. Governance *on*: k=2 evidence gate; selection *on*: pilot on the same arm, *off*: candidate-set mean. Brackets: 95% intervals; bold: best cell per column.

		Success (%)			
governance	order selection	MBPP	HumanEval+	LiveCodeBench	mean
off	off	43.71	70.60	32.99	49.10
off	on	43.97	**73.33**	**35.54**	**50.95**
on	off	44.03	69.67	33.05	48.92
on	on	**45.53**	72.53	33.38	50.48
governance main effect (pt)		$+0.94$	-0.87	-1.05	-0.33 $[-0.84, +0.26]$
order-selection main effect (pt)		$+0.88$	$+2.80$	$+1.44$	$+1.71$ $[+1.33, +2.05]$
interaction (pt)		$+1.24$	$+0.13$	-2.22	-0.28
order − governance (pt)		-0.06	$+3.67$	$+2.49$	$+2.04$ $[+1.43, +2.61]$

Three further readings show where the two levers overlap. The best LiveCodeBench cell is the ungoverned bank with a chosen order (35.54), above our complete system (33.38); on HumanEval+ our writer sits 0.93 points below the bank that admits everything; and on LiveCodeBench the interaction is -2.22, so there the levers compete for the same enrichment rather than compounding. Six further write- and retrieval-side interventions—learned gating, dosage control, family blocking, sticky refusal, retrieval filtering—fall between -0.39 and $+0.81$ points against their own controls, none significantly positive, so even the best, picked with hindsight, stays below the pilot's held-out $+1.71$ (Appendix E, Table 10). Finally we spend information no deployed system will have. For each candidate entry e we estimate its effect β_e from runs on seeds disjoint from the evaluation and admit greedily under a budget matched to random admission:

$$\mathcal{A}^\star \;=\; \arg\max_{\mathcal{A}\subseteq\mathcal{E},\,|\mathcal{A}|\le B} \sum_{e\in\mathcal{A}} \widehat{\beta}_e, \qquad \widehat{\beta}_e \;=\; \mathbb{E}\big[\,y \mid e \in \text{bank}\,\big] - \mathbb{E}\big[\,y \mid e \notin \text{bank}\,\big]. \tag{2}$$

This estimated-effect rule scores -0.27 points $[-0.78, +0.15]$ against random admission. $\widehat{\beta}_e$ is a conditional contrast over bank membership, and Eq. 2 sums per-entry marginals, so complementarity, bank-state conditioning and rewriting lie outside it; within this class, admitting on per-entry

Table 3: **Main result: held-out success with order selection off and on.** Seeds 5–9 (no memory: 3–4); gain in points; †: 95% interval contains zero. *off*: mean over the twelve candidate orders; *on*: each policy's own pilot pick (seeds 0–2). Baselines are in-harness replicas.

	MBPP			HumanEval+			LiveCodeBench		
	Success (%)		Gain	Success (%)		Gain	Success (%)		Gain
write policy	off	on	(pt)	off	on	(pt)	off	on	(pt)
memory controls									
no memory	43.74	43.97	+0.24	67.28	67.33	+0.06†	34.36	34.62	+0.26
ungoverned bank	43.71	43.97	+0.26†	70.60	73.33	+2.73	32.99	35.54	+2.55
write policies									
ours (governed)	44.03	45.53	+1.50	69.67	72.53	+2.87	33.05	33.38	+0.33†
Reflexion	43.81	45.53	+1.73	70.71	73.87	+3.16	33.58	36.00	+2.42
ExpeL	43.50	46.10	+2.60	70.78	73.07	+2.29	33.15	33.38	+0.23†
AWM	43.53	43.55	+0.01†	70.93	73.60	+2.67	33.41	34.15	+0.74†

effect estimates returns no gain (the widest interval, family blocking on Mercury, reaches +1.72). Across four interventions interception strength and gain also correlate negatively (−0.218, −0.319, −0.450, −0.417), so in all four a tighter gate goes with a lower gain. Table 2 carries the comparison of levers directly.

4.2 Order changes what the bank holds

The bank logs show why the two levers need not coincide: order changes which notes exist. Across the twelve orderings of one MBPP pool the round at which the writer consolidates its first durable skill moves by 25.8, the retrieval rate by 23 points and the token footprint nearly four-fold; on HumanEval+ it moves by 20.2 rounds and retrieval by 40 points; on LiveCodeBench consolidation moves by 50 of 110 rounds (Appendix F, Table 11). An admission rule therefore meets a different set of experiences under each order, so a single rule would have to suit every order at once to match the upstream choice. Periodic reset points the same way: emptying the bank every twenty rounds *enlarges* the ten-seed spread by +2.41 [+1.63, +3.05] on MBPP and +2.31 [+0.38, +3.15] on LiveCodeBench, so order matters within every window, not only through a long accumulated history. An early exploratory control separates part of the alternative: holding the bank fixed and shuffling where retrieved entries sit in the prompt flips 2.7% of reachable MBPP+ tasks, against 18.6% when the bank itself is swapped mid-stream; on HumanEval+ they are 5.5% and 6.5% (Appendix F). The largest case makes the chain concrete: on LiveCodeBench under our writer, hard-first consolidates its first durable skill at round 78.7 and a random permutation at round 28.7, with retrieval firing on 36% against 70% of steps. Read together, the two subsections say where the leverage sits. The order decides which experiences the writer ever produces, while admission can only rank the ones a given order has already produced. A gate tuned for one arrival sequence meets a different set of candidates under the next, so the same rule can be right for one sequence and wrong for another, whereas the sequence itself is settled once, before the first entry exists. Neither lever substitutes for the other, and only one of them is fixed before the run begins, which is where a deployer can still act. Admission is itself part of the loop: what the gate keeps steers later solutions and so the experiences that follow. Table 2 scores every rule over complete runs, so that downstream influence is already inside the governance effect it reports.

5 The ordering lever

5.1 Protocol and gains

Since the leverage lies in the order, a deployer needs a way to pick one for a given pool. We formalise order selection as a mechanism that attaches to *any* write policy and leaves it untouched. Let S_{pilot} be three pilot seeds and S_{test} five held-out seeds. With the mechanism off a deployer does not choose and earns the mean over candidate orders; with it on the deployer commits to the pilot winner, and

the gain is the difference:

$$\widehat{o} \;=\; \arg\max_{o\in\mathcal{O}} \frac{1}{|S_{\text{pilot}}|} \sum_{s\in S_{\text{pilot}}} y(o,s), \qquad G \;=\; \frac{1}{|S_{\text{test}}|} \sum_{s\in S_{\text{test}}} y(\widehat{o},s) \;-\; \frac{1}{|\mathcal{O}|} \sum_{o\in\mathcal{O}} \overline{y}(o). \tag{3}$$

The pilot costs $3|\mathcal{O}| = 36$ trials, 60% of the 12×5 grid we spend to evaluate a single pool, paid once per pool rather than once per run. It adds no tasks to label, no model to train and no change to the write policy it fronts, and is amortised over every later run of that pool—the regime of §2.

Gains across writers. Table 3 reports absolute held-out success with the pilot off and on, for four write policies and both memory controls. Eight of twelve policy intervals exclude zero, and all four gain significantly on HumanEval+: Reflexion $+3.16$ $[+2.04, +4.13]$, ours $+2.87$ $[+2.02, +3.56]$, AWM $+2.67$ $[+2.36, +3.04]$, ExpeL $+2.29$ $[+1.62, +2.80]$. Three of the four largest cells belong to literature baselines (Reflexion $+3.16$, ExpeL $+2.60$, AWM $+2.67$), so the gain attaches to the loop and not to our writer. The control rows fix the scale both ways. Without a bank the pilot buys at most $+0.26$ points anywhere, so it is not finding an easier curriculum; and the six *off* configurations sit within 0.53 points on MBPP, where the choice of writer is thus worth a fraction of the choice of order.

Information boundary. Eq. 3 selects on y. Within a pool the pilot isolates *seeds*: it scores candidates on the held-out set the final number uses, with seeds disjoint from those that report it, and fits no parameters. This is model selection on a fixed, repeatedly graded workload (§2), and Table 3 reports what a pool-internal search earns. The 34 fresh pools below test whether the *procedure* carries to new pools.

5.2 Copying versus re-measuring

A good order is a sequence×writer interaction—the sign flip of §3.2 from the other side—so it does not travel, and Table 4 makes that boundary quantitative by putting both options in one row. Copying the order our pilot chose onto a baseline writer is significantly harmful in seven of nine cells and significantly helpful in none; letting that same writer spend the same pilot on its own arm helps significantly in six, with a positive estimate in all nine. Averaged over the nine cells, copying returns -1.27 points and the writer's own pilot $+1.76$. Nor is the order predictable from the task set: a ridge regression from family statistics, leave-one-variant-out, scores -0.34 points on MBPP, and cross-pool rank transfer stays near zero (Spearman -0.027 and $+0.134$). The protocol also supplies the operational rule Ye et al. (2026) stop short of—report multiple runs, *and* re-measure the order on the pool you will run.

Table 4: **Applicability: copy the order, or run the pilot yourself?** Gains (pt) over each writer's candidate-set mean (*base*), seeds 5–9; †: 95% interval contains zero. *copy*: our writer's pilot order; *own*: the writer's own pilot.

	MBPP			HumanEval+			LiveCodeBench		
	base	Gain (pt)		base	Gain (pt)		base	Gain (pt)	
receiving writer	(%)	copy	own	(%)	copy	own	(%)	copy	own
Reflexion	43.81	-0.97	$+1.73$	70.71	$-1.11^{\dagger}$	$+3.16$	33.58	-1.42	$+2.42$
ExpeL	43.50	-0.80	$+2.60$	70.78	-2.51	$+2.29$	33.15	-1.77	$+0.23^{\dagger}$
AWM	43.53	-0.98	$+0.01^{\dagger}$	70.93	-2.13	$+2.67$	33.41	$+0.28^{\dagger}$	$+0.74^{\dagger}$

5.3 When to trust the pick

The pilot's pick is trustworthy when the order ranking inside a pool is stable across seeds, and that stability can be read off the pilot runs themselves, using no seed the final number is scored on. Writing $\overline{y}_A(o)$ for the mean of order o over seed set A, the statistic averages a leave-one-seed-out correlation over the pilot seeds:

$$\widehat{\rho} \;=\; \frac{1}{|S_{\text{pilot}}|} \sum_{s\in S_{\text{pilot}}} \operatorname{corr}\Big(\overline{y}_{\{s\}}(\cdot),\; \overline{y}_{S_{\text{pilot}}\setminus\{s\}}(\cdot)\Big), \qquad \text{adopt if } \widehat{\rho} \geq 0.70, \;\; \text{decline if } \widehat{\rho} \leq 0.45. \tag{4}$$

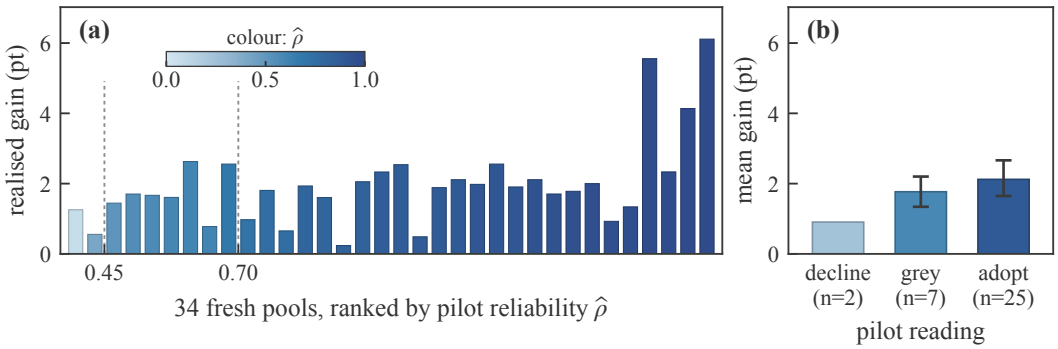


Figure 4: **The pilot's own reliability tracks its payoff.** (a) Gain on seeds 3–4 for 34 fresh pools, ranked and coloured by pilot $\widehat{\rho}$ (seeds 0–2); dotted: pre-registered thresholds. (b) Mean gain per reading, 95% bootstrap interval (none for n=2).

The correlation is Pearson's and both thresholds were fixed in a pre-registration dated before the HumanEval+ fresh-pool run; between them the reading is grey and licenses no claim. It is read after the pilot has run, to decide whether to commit. On the four main pools it reads 0.844, 0.697 and 0.592 on MBPP, HumanEval+ and LiveCodeBench, and the same correlation on the untouched seeds 5–9 reads 0.950, 0.662 and 0.348, the same ranking (Table 12); only MBPP clears 0.70, and HumanEval+ sits at the threshold. The real test is prospective. On the 34 fresh pools, $\widehat{\rho}$ tracks the realised gain pool by pool (Figure 4; Spearman $+0.46$): the 25 pools read as *adopt* gain $+2.12$ $[+1.64, +2.69]$, the seven grey pools $+1.77$ $[+1.34, +2.20]$, and the two read as *decline* $+0.91$. Every fresh pool gains, so a low reading signals a smaller payoff rather than a loss, and the pick ties the best order in 28 of the 34 pools (Table 5). Mercury, excluded before unblinding, reads grey and, scored post hoc, gains $+2.40$ $[+1.48, +3.32]$. On every benchmark the fresh-pool gain also exceeds the governance main effect of Table 2: $+1.85$ against $+0.94$ on MBPP, $+2.26$ against -0.87 on HumanEval+ and $+1.88$ against -1.05 on LiveCodeBench.

Cheaper variants. Two variants bracket Table 5. The pre-registered two-seed pilot, a third cheaper at 24 trials, returns $+1.53$, $+2.48$ and $+2.07$ points on the three families (84, 99 and 99% of the ceiling). Restricting candidates to the four orders that need no reference solution returns $+1.23$ $[+0.89, +1.61]$, $+2.08$ $[+1.20, +3.07]$ and $+1.56$ $[+0.98, +2.26]$: reference-solution orders widen the harvest but are not its source. On LiveCodeBench the main pool reads grey and gains $+0.33$ $[-1.09, +1.76]$, while nine of twelve resampled pools read *adopt* and gain $+1.88$: the reading follows the pool, not the benchmark (Appendix H lists all 34).

Table 5: **Criterion: the full protocol rerun on 34 fresh pools.** Pilot $\widehat{\rho}$: Eq. 4 on seeds 0–2, mean over pools; *reading*: adopt/grey/decline counts; gain on seeds 3–4; *ceiling*: share of the best-order gain recovered; *best*: pools where the pick ties the optimum.

pool family	pools	pilot $\widehat{\rho}$	reading	gain (pt)	ceiling	best
MBPP, resampled	12	0.775	10/1/1	$+1.85$ $[+1.34, +2.40]$	97%	11/12
HumanEval+, split	10	0.789	6/3/1	$+2.26$ $[+1.49, +3.26]$	93%	9/10
LiveCodeBench, resampled	12	0.812	9/3/0	$+1.88$ $[+1.31, +2.66]$	91%	8/12

Scale and transfer. The doubled MBPP universe of §3 also preserves the gain: on its 299 training and 181 held-out problems the pilot returns $+2.19$ $[+1.24, +2.99]$, above the $+1.50$ on the original pool. Going from five to ten held-out seeds leaves the sign of all twelve gain cells unchanged. With the memory subsystem of ReasoningBank (Ouyang et al., 2026) ported into the harness, the order effect persists (spread 4.86 points) and the pilot returns $+0.95$ on a bank $9.33\times$ larger than ours.

Scope of the claim. The claim concerns *where* governance has leverage: the six *off* configurations lie within 0.53 points on MBPP (Table 3), and §4 covers the admission mechanisms we built. The comparison is in accuracy at identical per-run cost; governance may still win on storage, latency

or auditability. The literature writers keep each method's entry type and write trigger but share our retrieval and prompting (Appendix J, Table 6), so the evidence covers their write rules under a shared harness. All confirmatory contrasts use execution-graded code on one backbone family; extending the pilot to live request streams and dependency-constrained workflows is the next step.

6 Related work

Memory in self-evolving agents. Memory-based self-evolving agents are organised around the two decisions downstream of the write: what to store—verbal failures (Shinn et al., 2023), distilled insights (Zhao et al., 2024), workflows (Wang et al., 2024b), reasoning strategies (Ouyang et al., 2026), trajectory exemplars (Zheng et al., 2024), long-lived stores (Xu et al., 2026; Chhikara et al., 2025; Packer et al., 2023; Zhong et al., 2024), skill libraries (Wang et al., 2023)—and how to retrieve it, inheriting retrieval-augmented generation (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020; Gutiérrez et al., 2024; Asai et al., 2024). We hold these writers fixed, three as baselines, and move upstream to *when* the stream reaches the writer at all.

Order effects in learning. When models learn by gradient updates, presentation order is well studied: curriculum learning ranks examples by difficulty (Bengio et al., 2009; Kumar et al., 2010; Graves et al., 2017; Wang et al., 2021), pretraining corpora are sequenced deliberately (Xie et al., 2023; Chen et al., 2023), and forgetting and replay are studied as its failure mode (Kirkpatrick et al., 2017; Fedus et al., 2020). What is new is the same sensitivity with *no* weight updates, carried by a textual store the agent writes for itself—which is why the no-memory control of §3.2 is load-bearing. Voyager's automatic curriculum (Wang et al., 2023) chooses *which* tasks to attempt; we fix the multiset and vary only its permutation. Two nearer phenomena must be separated from ours: position inside one context window (Lu et al., 2022; Zhao et al., 2021; Liu et al., 2024a), which can act through what the bank retrieves and which an exploratory control finds small on MBPP+ (Appendix F); and primacy bias in reinforcement learning (Nikishin et al., 2022; Sokar et al., 2023), the closest analogue to a bank that locks in early, which here appears as a 25.8-round shift in when the first durable skill forms (Appendix F). The pilot itself is uniform-allocation candidate selection; adaptive schemes such as successive halving (Jamieson & Talwalkar, 2016; Li et al., 2018) could cut its 36 trials.

Order in evaluation. Ye et al. (2026) are the closest prior result and our motivation: default orders act as an implicit curriculum, and their prescription is stricter multi-run reporting. We localise the effect in the memory pathway and turn the diagnosis into an instrument with an applicability test. Code (Austin et al., 2021; Chen et al., 2021; Liu et al., 2023; Jain et al., 2025; Du et al., 2024; Hendrycks et al., 2021; Lai et al., 2023; Zhuo et al., 2025) and agent benchmarks (Liu et al., 2024b; Jimenez et al., 2024; Yang et al., 2024; Wang et al., 2024a) alike report one run on a canonical stream, so the pilot is also a reporting discipline: state the order, and re-measure it for each writer.

7 Conclusion

Self-evolving agents are usually governed downstream, when an experience is admitted. Put on the same footing—same arms, seeds and per-run cost—the upstream decision returned $+1.71$ points where our governance returned -0.33, and no further admission intervention, an estimated-effect rule among them, gained significantly. The bank shows why the two levers differ: reordering the same tasks changes which experiences exist, so an admission rule meets a different set under every order. The gap is widest on HumanEval+, where order selection returned $+2.80$ points and governance -0.87. The pilot that makes the choice recovered 91–97% of the best-order gain on 34 fresh pools, and its reliability reading ranked those pools by payoff.

The results support two rules. For evaluation: *when a system writes its own experience into its own memory, arrival order is an independent variable, not noise*. For deployment: *measure the order on the pool and with the writer you will run, and read the pilot's reliability before committing*—our winner hurt seven of nine other writers, while their own pilots helped six. Admission is the visible decision, but it acts on experiences the stream already produced, and that stream is free to permute before a single entry is written.

AI USE STATEMENT

AI assistance was used in drafting this article and in the retrieval and discovery of related work. All research concepts, ideas and analyses were developed and conducted by the authors, and all editorial judgement and conclusions are the authors' own. The authors take full responsibility for the content of this paper.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. No human subjects or animal experimentation were involved. All datasets used, namely MBPP, HumanEval+, LiveCodeBench and Mercury, are public code-generation benchmarks and were used in accordance with their release terms; all backbones are publicly released open-weight Qwen2.5 models. The tasks are programming problems graded by test execution and contain no personally identifiable information or demographic attributes. Memory entries hold only the agent's own traces on training tasks and exist inside the evaluation harness; no user-facing system was deployed, and no experiment raises privacy or security concerns.

REPRODUCIBILITY STATEMENT

The experimental setup, including the backbone, decoding temperature, benchmarks, train/held-out draws, the twelve orderings and the seed split, is described in §2. The estimands are defined by Eq. 1, Eq. 2 and Eq. 3, and the reliability reading by Eq. 4. Appendix J specifies the experimental units, the pre-registration dates and falsification thresholds (Table 15), the resampling procedures and the reimplementation of the three literature writers. The supplementary material contains the anonymised source code of the harness, the suite manifests that freeze every task draw and ordering, the pre-registration documents, the per-unit raw results underlying every table and figure, and the analysis scripts that regenerate them. All four benchmarks are publicly available, so every evaluation can be rerun on identical task sets. We will release the code and data publicly upon acceptance.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. volume 34, pp. 29304–29320, 2021.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *Advances in Neural Information Processing Systems*, 38:113506–113543, 2026.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. pp. 41–48, 2009.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. Accounting for variance in machine learning benchmarks. *Proceedings of machine learning and systems*, 3:747–769, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Mayee Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models. *Advances in Neural Information Processing Systems*, 36:36000–36040, 2023.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. Show your work: Improved reporting of experimental results. pp. 2185–2194, 2019.

Mingzhe Du, Luu A Tuan, Bin Ji, Qian Liu, and See-Kiong Ng. Mercury: A code efficiency benchmark for code large language models. volume 37, pp. 16601–16622, 2024.

Bradley Efron. Bootstrap methods: another look at the jackknife. pp. 569–593, 1992.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International conference on machine learning*, pp. 3061–3071. PMLR, 2020.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. pp. 1311–1320, 2017.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. volume 37, pp. 59532–59569, 2024.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. pp. 3929–3938, 2020.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. 2021.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*, pp. 240–248. PMLR, 2016.

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*, volume 2024, pp. 54107–54157, 2024.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. volume 23, 2010.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. Ds-1000: A natural and reliable benchmark for data science code generation. pp. 18319–18345, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. volume 33, pp. 9459–9474, 2020.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of machine learning research*, 18(185):1–52, 2018.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. volume 36, pp. 21558–21572, 2023.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024a.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024b.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, 2022.

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. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Evgenii Nikishin, Max Schwarzer, Pierluca DOro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pp. 16828–16847. PMLR, 2022.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. 2023.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. pp. 1–22, 2023.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *Journal of machine learning research*, 22(164):1–20, 2021.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. volume 36, pp. 8634–8652, 2023.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 32145–32168. PMLR, 2023.

Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

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.

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576, 2021.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. 2024a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. 2022.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. 2024b.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. volume 36, pp. 69798–69818, 2023.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. volume 38, pp. 17577–17604, 2026.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. volume 36, pp. 11809–11822, 2023.

Qinyuan Ye, Yu Li, Yada Pruksachatkun, Jiaxin Zhang, and Chien-Sheng Wu. On the fragility of self-improving agents: Variance, task order, and underspecification. *arXiv preprint arXiv:2608.18066*, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pp. 12697–12706. Pmlr, 2021.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations*, volume 2024, pp. 19036–19066, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. 38(17):19724–19731, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. 2022.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, volume 2025, pp. 66602–66656, 2025.

A EXTENDED RELATED WORK

§6 places this paper against its neighbours in the space a conference page allows. This section gives each neighbour the room to be stated on its own terms, and says in each case what the present result adds and what it does not touch.

What a memory-based self-evolving agent decides. Strip the systems back and each one is a choice on two axes. The first is what a completed episode leaves behind: a verbal post-mortem of a failure (Shinn et al., 2023), an insight distilled across a batch of episodes (Zhao et al., 2024), a reusable workflow induced from repeated successes (Wang et al., 2024b), a reasoning strategy (Ouyang et al., 2026), a trajectory kept as an exemplar (Zheng et al., 2024), an executable skill (Wang et al., 2023), or a durable personal store (Packer et al., 2023; Zhong et al., 2024; Xu et al., 2026; Chhikara et al., 2025). The second is how the store is reached at inference time, which the field largely inherits from retrieval-augmented generation (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020) and its later refinements (Gutiérrez et al., 2024; Asai et al., 2024). Sumers et al. (2023) organise these choices into a single architecture, and Behrouz et al. (2026) pursue the same objective in weights rather than in text. Both axes act *after* an episode exists. The stream that decides which episodes exist at all is the axis this paper varies, and it is orthogonal to every choice above: the three writers we reimplement, listed with their exact triggers in §J, each meet a different set of episodes under a different arrival order, which is why Table 4 finds one order does not transfer between them.

Order in gradient-based learning. When learning is a sequence of weight updates, presentation order is a first-class design variable. Curriculum learning orders examples from easy to hard (Bengio et al., 2009), self-paced variants let the model set that ordering (Kumar et al., 2010), and an automated teacher can select it online (Graves et al., 2017); Wang et al. (2021) survey the family. At pretraining scale the same question becomes domain mixing and scheduling (Xie et al., 2023; Chen et al., 2023). The failure mode has its own literature: what arrives late overwrites what arrived early (Fedus et al., 2020), and the standard remedies constrain the update itself (Kirkpatrick et al., 2017). All of this presumes a gradient. The setting here has none—the weights are frozen at temperature 0 and the only thing that persists between tasks is text the agent wrote—and that is what makes the no-memory control of §3.2 load-bearing rather than decorative: with the store disabled the spread falls to the inference floor, so the carrier is the store and not the backbone.

Position effects inside a single context. A nearer relative is the sensitivity of a single forward pass to where material sits in its prompt: the order of in-context demonstrations moves accuracy sharply (Lu et al., 2022), the resulting bias is partly correctable by calibration (Zhao et al., 2021), and evidence placed mid-context is used less than evidence at either end (Liu et al., 2024a). This is a different mechanism operating on a different timescale—one prompt rather than a run of 240 tasks—but the two can compose, because what the store retrieves is itself laid out in a prompt. §F reports the exploratory control that separates them on MBPP+.

Primacy in reinforcement learning. The closest analogue to a bank that locks in early is the primacy bias of deep reinforcement learning, where early experience disproportionately shapes what is learned afterwards (Nikishin et al., 2022) and units can stop contributing altogether (Sokar et al., 2023). The shape of the phenomenon matches: an early commitment that later data does not undo. The substrate does not. Here the commitment is a set of text entries with an explicit admission rule and an explicit eviction rule, both inspectable, which is what allows §F to date it—the first durable skill forms 25.8 rounds apart between the extreme orderings—rather than infer it from behaviour.

Variance and reporting discipline. A recurring finding in empirical machine learning is that headline differences are smaller than the variation of the pipeline that produced them, in deep reinforcement learning (Henderson et al., 2018; Agarwal et al., 2021), in reported natural-language results (Dodge et al., 2019), and across benchmark protocols generally (Bouthillier et al., 2021); Pineau et al. (2021) turn the observation into a checklist. The order effect belongs to this literature as a source of variation that current protocols do not report at all: a system that writes its own experience and is evaluated on one canonical stream has one draw from a distribution whose spread this paper measures at 1.71 to 6.13 points depending on the pool. Our response is the one this literature recommends—thresholds frozen before unblinding, a confirmatory seed block never read during

development, and intervals clustered on the ordering—and the documents are listed in §J with the predictions they committed to in Table 15.

B EXPERIMENTAL SETUP IN FULL

This section fixes every knob of the harness that §2 summarises. Table 6 lists the configuration. It is identical across all confirmatory arms, and each pre-registration document of §J copies it verbatim, so that a later experiment stays paired with the published grid and differs from it in exactly one factor.

Table 6: **Harness configuration, identical across arms.** Each row is one setting and the value it was frozen at, grouped by backbone, store, retrieval and protocol. The only factors that vary between experiments are the ones a given contrast is about: the arrival order, the write policy, the backbone and the task pool.

setting	value
backbone	
model (confirmatory)	`Qwen2.5-14B-Instruct`
decoding temperature	0
generation budget	900 tokens
store	
entry levels	3 (memory, skill, rule)
token budget per level	900
consolidation similarity τ	0.25
failed attempts written	no
retrieval	
entries per prompt	3
retrieval token budget	200
similarity floor	0.05
protocol	
orderings per stream	12
held-out split seeds (unioned)	$0, 1, 2$
pilot seeds / scoring seeds	0–2 / 5–9
write-gate evidence threshold	2 solved tasks

Benchmarks and pools. Each benchmark contributes one task universe, sampled once and frozen in a manifest: 240 problems from MBPP, 160 from HumanEval+ (the EvalPlus release holds 164, so this is very nearly the whole set), 240 from LiveCodeBench and the Mercury set. A pool is then split into a training stream and a held-out set by drawing up to 25 held-out tasks per task family under each of three split seeds and taking the union, which gives the $99/141$, $85/75$ and $110/130$ splits of §2. Pooling three splits rather than trusting one matters here: a single split is too coarse to resolve differences of one or two points, which is the size of the effects this paper reports. The same manifest is reused by every arm and every ordering, so two runs that differ in arrival order face a bit-identical task set.

The twelve orderings. Eight orderings are random permutations of the training stream drawn from a fixed order seed, so that every arm sees the same eight. The remaining four are constructed. *Easy-first* and *hard-first* sort by the length of the reference solution, the one difficulty signal available offline without running the agent; they are the two orderings that assume an offline pool, and §5 reports the pilot restricted to the ten that do not. *Blocked-by-family* groups tasks of the same family into contiguous runs, and *interleaved* does the opposite, rotating through the families so that consecutive tasks rarely share one. The four constructed orderings bracket the random ones in Figure 2(a) rather than dominating them, which is why the pilot searches over all twelve instead of defaulting to a difficulty heuristic.

The store. Entries live at three levels. A *memory* entry records the task statement and the verified solution; a *skill* entry records the solution under a family tag; a *rule* entry names the family, the goal and the algorithmic shape, and drops the code. All three are produced by deterministic templates,

so no arm buys extra inference compute by writing, and a comparison between routing policies measures routing rather than generation. Only solved tasks are written. A candidate is suppressed when the store already holds an entry of the same level and family whose IDF-cosine similarity to it reaches $\tau = 0.25$; this is the consolidation rule whose timing Table 11 measures. Each level holds its own 900-token budget and evicts oldest-first on overflow.

Retrieval and prompting. Before each task the store scores every entry by IDF-cosine similarity against the task statement, keeps those above 0.05, and fills the context with the best three under a 200-token budget. The agent then sees one system message and one task message, both fixed for every arm and every run and both reproduced in §I. A candidate is accepted only if a fenced block defines `solution`; it is then executed in a separate process against the official unit tests, so that a wrong answer can neither hang nor crash the driver, and grading stays binary.

Held-out scoring. A run ends when the training stream is exhausted, and only then is the held-out set touched. The store is closed before scoring begins: writes are disabled, and the driver asserts a hash of the store's contents before and after the held-out pass, so a scoring bug that mutated the bank would abort the unit rather than quietly inflate it. Every held-out task is then attempted once, retrieval still active, and graded binary against the benchmark's official unit tests in a separate process. Two consequences matter for the contrasts in this paper. First, the number the tables report is a frozen-bank number, so an ordering can only help by leaving behind a better bank, not by being luckier during the stream. Second, because the held-out set is fixed by the manifest, two orderings of the same pool are scored on a bit-identical set of tasks, and every difference between them is paired at the task level.

Seeds and what a seed measures. Decoding is at temperature 0, so re-running a seed replays an identical configuration and seed-to-seed variation is the nondeterminism of batched inference on the GPU rather than sampling. That floor is not assumed, it is measured: the no-memory arm runs the same twelve orderings with writes and retrieval disabled, and whatever spread it shows is the spread attributable to the hardware. Ten seeds are budgeted per confirmatory pool and split by role before anything is read. Seeds 0–2 fit, seeds 3–4 confirm during development, and seeds 5–9 are held in reserve for a single confirmatory judgement; the pilot of §5 selects on the first block and is scored on the last, so the number it earns is out-of-sample by construction and no resampling correction is needed to make it so.

What a run costs. The comparison of §4 is between arms that spend the same compute, and Table 7 shows what that means at the level of generator calls. Permuting a stream cannot change the count, since the agent still attempts every task exactly once; and every arm in the 2×2 of Table 2—no memory, the ungoverned bank, the evidence gate, our governed writer, periodic reset—makes the identical number of calls on all four benchmarks. The three literature writers spend *more*: between 1% and 28% extra, because each buys an additional reflection, insight-extraction or workflow-induction call. The head-to-head of §5 therefore gives them the larger budget, not the smaller one.

Two costs sit outside a single run. The pilot is one of them: twelve candidate orders on three seeds is 36 trials, or 24 in the two-seed variant of §5, paid once for a pool and amortised over every later run on it, which is the regime §2 defines. The other is the search over admission rules, and we spent it on governance rather than against it: §4 reports seven further rules, one of them estimating per-entry effects from runs a deployed system could never see, and the best of the seven, chosen after the fact, still falls below the pilot's held-out $+1.71$.

C Pool geometry and the twelve decision units

Table 1 gives both readings of the order effect referenced in §3. *Spread* is the exposure of an unreported choice, while *held-out best−worst* is what a selection procedure with no winner's curse would see, and only the latter is on the same footing as the gains in Table 3. The three streams per benchmark are the replication unit for spread, so we report their range rather than a bootstrap interval; held-out intervals are 95% seed-clustered bootstrap intervals. Binary execution grading

Table 7: **Generator calls per run, by arm.** Mean over the twelve orderings and ten seeds (five for no memory and the literature writers). Permuting a stream leaves the count unchanged by construction, and every arm of Table 2 matches the ungoverned bank exactly; the literature writers each buy one extra call per reflection, insight or workflow.

arm	MBPP	HumanEval+	LiveCodeBench	Mercury
no memory	240	160	240	179
ungoverned bank	240	160	240	177
evidence gate	240	160	240	177
ours (governed)	240	160	240	177
periodic reset	240	160	240	177
ExpeL	244	164	245	184
AWM	249	172	248	192
Reflexion	291	181	307	208

fixes what one test case is worth, so every effect converts into tasks. Mercury is pre-registered as descriptive (§2).

Table 8 opens the twelve units behind that summary. Each row is one benchmark crossed with one independently sampled stream, and the permutation test shuffles order labels within each seed. The spread runs from 4.1 to 10.7 points and survives the Holm correction in eleven of the twelve; the exception is the single Mercury stream on which the ranks were least stable, which is why Mercury is reported descriptively throughout.

Table 8: **The twelve decision units.** *Success*: grand mean over the twelve orderings. *Spread*: best−worst of the twelve. p: permutation test on order labels, shuffled across all runs of the unit, $20{,}000$ draws. *Holm*: adjusted over the twelve units. †Not significant at 0.05.

benchmark	stream	success (%)	spread (pt)	p	Holm p
MBPP	s0	48.5	7.6	< 0.001	< 0.001
	s1	47.6	7.3	< 0.001	< 0.001
	s2	60.5	8.3	< 0.001	< 0.001
HumanEval+	s0	62.7	10.7	< 0.001	< 0.001
	s1	63.6	6.4	< 0.001	0.002
	s2	73.4	7.9	< 0.001	< 0.001
LiveCodeBench	s0	31.3	4.1	0.002	0.005
	s1	34.5	5.1	0.021	0.043
	s2	37.7	5.5	< 0.001	< 0.001
Mercury	s0	54.5	9.4	< 0.001	0.001
	s1	69.5	5.3	0.076	$0.076^{\dagger}$
	s2	61.2	10.3	< 0.001	< 0.001

A permutation test says the spread is not zero; Figure 5 says how far from zero it is. For each benchmark and each bank we rebuild the null by shuffling order labels within seed and recording the spread this produces by chance alone. The observed spread clears the null's 95th percentile in all eight cells, by a factor of 1.1 to 2.1, and the ordering of the benchmarks is the same under both banks, so the geometry of the pool, not the write policy, sets how much room an order has to move the outcome.

D BACKBONE SCALE, POOL SIZE AND MEMORY IMPLEMENTATION

Three registrations ask whether the effect of §3 is an artefact of the particular backbone, the particular pool size or our own store. Each was frozen with a falsification threshold before any of its results were read, and Table 9 collects what they returned.

Across the Qwen2.5 line the spread never falls below 3.6 points and the curve does not decay with capability: on MBPP the smallest value sits at 14B and rises again at 32B, while on HumanEval+ the four scales lie within 0.53 points of one another. Tripling the training stream does not drain

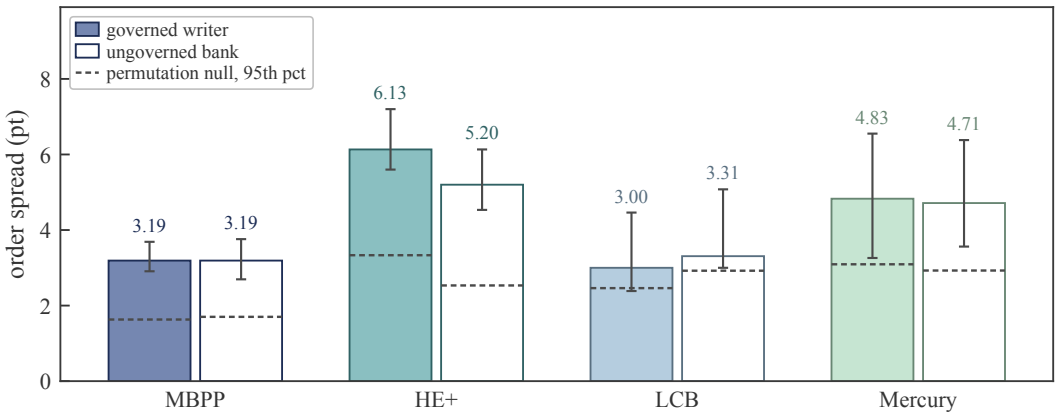


Figure 5: **The order spread against the spread that chance alone produces.** Bar height and printed value: observed spread, for the governed writer (filled) and the ungoverned bank (open); whisker: 95% seed-clustered bootstrap interval; dashed rule inside each bar: 95th percentile of the permutation null, which shuffles order labels within each seed. Every bar clears its own null.

the effect either: the doubled MBPP universe carries a spread of 5.19 points where the original pool carried 3.69, and the pilot returns $+2.19$ $[+1.24, +2.99]$ on it. With the memory subsystem of ReasoningBank (Ouyang et al., 2026) in place of our own store, on a different backbone and a bank $9.33\times$ larger, the spread is 4.86 points and the pilot still returns $+0.95$.

What the ReasoningBank port keeps. That last row is worth unpacking, because it is the row that answers whether the effect belongs to our own writer. The port takes the memory module of Ouyang et al. (2026) and carries four things across byte for byte: the two distillation system instructions, the distillation temperature and the at-most-three-items output contract, the write-everything semantics under which successful and failed trajectories are both distilled and appended with no admission filter and no capacity bound, and the retrieval path, which prefixes the instruction string to the query, scores cosine similarity against the cached embeddings of past queries and takes the top n. The embedding checkpoint and the distilling model are the ones our endpoint serves, with the mean-pool-over-attention-mask and L_2 normalisation of the original kept line for line. Six orderings and five seeds run on this arm, on MBPP with the same $99/141$ split, and the spread of 4.86 points in Table 9 is measured on a bank that ends $9.33\times$ larger than ours. The effect survives a different writer, a different retriever and a different backbone at once.

One reading of these cells deserves separating from the other. A stronger backbone solves more of the pool outright, and a task solved under every ordering cannot contribute to a spread, so the room an ordering has to move the outcome is bounded above by the share of held-out tasks whose verdict is not constant across all sixty runs. That share does fall with capability—on MBPP it goes from 26.2% at 7B to 23.4% at 14B and 17.7% at 32B, and on HumanEval+ from 36.0% to 22.7% and 18.7%—and the spread does not follow it down. The effect is therefore not a fixed fraction of the available room: as the room narrows, the ordering claims more of what remains. This is the cell we would point a reader to who suspects the phenomenon is a small-model artefact, because it is the one place where the two quantities move apart.

E THE GOVERNANCE PANEL

Table 10 lists the six write- and retrieval-side interventions summarised in §4; the dosage ladder appears on two pools, and periodic reset is reported in the main text as a change in spread, not in accuracy. Each row is a separate experiment against its own matched control, so the rows are *not* a ranking: the benchmark, the control and the number of paired cells differ by row, and only the column of paired differences is comparable across them. All rows run on development seeds 0–4 by design, since confirmatory seeds 5–9 were reserved for §5 and never read while this space was explored.

Table 9: **The spread survives every change of backbone, pool size and store.** Each row is one pre-registered experiment. *Spread*: best−worst of the twelve orderings within a seed, averaged over seeds, on the ungoverned bank. The pre-registered floor was 2.0 points in every cell. *Six orderings and a different backbone, reported at its own grain.

level	pool	spread (pt)
backbone, twelve orderings, five seeds		
Qwen2.5-7B	MBPP	5.82
Qwen2.5-14B	MBPP	3.69
Qwen2.5-32B	MBPP	5.53
Qwen2.5-72B	MBPP	4.68
Qwen2.5-7B	HumanEval+	6.13
Qwen2.5-14B	HumanEval+	6.13
Qwen2.5-32B	HumanEval+	5.60
Qwen2.5-72B	HumanEval+	5.87
pool size, ten seeds		
240 tasks, 99/141 split	MBPP	3.69
480 tasks, 299/181 split	MBPP	5.19
memory implementation		
ReasoningBank module*	MBPP	4.86

Each row is one answer to "what should the bank keep". *Family blocking* refuses entries from a family the store already covers, the coarsest form of redundancy control. The *learned gate* replaces the fixed two-solution evidence threshold with a classifier trained on write-time features to predict whether an entry will later help, and the *sticky* variant makes its refusals persistent, so that an entry rejected once is not reconsidered. The *retrieval-side filter* moves the decision downstream, leaving the write rule alone and instead dropping retrieved entries that score poorly against the current task. The *dosage ladder* varies how much is written per solved task rather than which entries are written, which separates the amount of experience from its selection. Underneath all of them sits the compression axis: an entry can be stored as a full memory, as a tagged skill or as a rule that keeps the goal and the algorithmic shape and drops the code, and the router picks the level from the number of times its family has been seen. The panel searches this space in both directions—what to admit, how much to admit, at what fidelity, and whether to filter at write or at read time—and none of it reaches the $+1.71$ that reordering the same tasks returns.

The estimated-effect rule. The last row is the one that spends information no deployed system will have, and it is worth stating exactly what it is given. For every candidate entry e we run the agent on seeds disjoint from the seeds the row is scored on, record whether e was in the bank and whether the later task was solved, and take $\widehat{\beta}_e$ from Eq. 2 as that entry's estimated contribution. Admission is then greedy in $\widehat{\beta}_e$ under a write budget matched to random admission, so the rule and its control keep banks of the same size and the comparison is about *which* entries are kept, not how many. A deployer cannot run this rule: the estimates arrive only after the runs that the entries were supposed to improve. It is included as a ceiling on the family, and it scores -0.27 $[-0.78, +0.15]$. The reason is the one §3.2 gives: an entry's contribution is not a property of the entry. The same note is worth keeping under one arrival sequence and worth dropping under another, so an estimate averaged over sequences describes no sequence in particular.

F BANK CONTENT ACROSS ORDERS

Every run stores its final bank, and each entry records the round at which it was consolidated, the number of tasks supporting it, and its family. This lets us measure directly what §4 asserts: that reordering changes the experiences the writer produces, not only when they are used. Across the twelve orderings of one pool, holding task content and write policy fixed, the resulting banks differ substantially.

Table 10: **Write- and retrieval-side interventions, each against its own control.** Δ is a paired difference against a matched control at equal write budget, over n paired (ordering, seed) cells, with a 95% bootstrap interval clustered on the twelve orderings. The estimated-effect row admits on the estimated per-entry effect $\widehat{\beta}_e$ of Eq. 2, estimated on seeds disjoint from its evaluation.

intervention	pool	n	Δ (pt)	95% CI
family blocking	Mercury	60	$+0.81$	$[-0.08, +1.72]$
learned gate	HumanEval+	60	$+0.04$	$[-0.87, +0.84]$
learned gate + sticky	HumanEval+	60	$+0.11$	$[-0.73, +0.93]$
retrieval-side filter	HumanEval+	60	-0.33	$[-1.11, +0.36]$
dosage ladder	HumanEval+	60	-0.09	$[-0.91, +0.76]$
dosage ladder	MBPP	240	-0.39	$[-0.67, -0.14]$
estimated-effect admission	MBPP	60	-0.27	$[-0.78, +0.15]$

Table 11: **Arrival order moves what the bank holds, not only how well the agent scores.** Each cell is a spread across the twelve orderings—the largest order mean minus the smallest, each mean taken over ten seeds—for one property of the end-of-run bank: the round at which the writer first consolidates a durable skill, the token footprint, the share of steps on which retrieval fires, and the entry count. Same tasks and same write policy throughout; only the arrival sequence differs.

benchmark	writer	consol. round	tokens	retrieval (%)	entries
MBPP	ours (governed)	25.8	226	23.2	1.0
MBPP	ungoverned bank	17.5	54	14.1	12.6
HumanEval+	ours (governed)	20.2	197	39.8	2.1
HumanEval+	ungoverned bank	7.4	107	13.5	8.2
LiveCodeBench	ours (governed)	50.0	417	33.6	2.4
LiveCodeBench	ungoverned bank	23.0	134	18.7	6.8

The largest cell is the clearest: on LiveCodeBench under the governed writer, hard-first consolidates its first durable skill at round 78.7 and one random permutation does so at round 28.7, half the run apart, with retrieval firing on 36% of steps against 70%.

We also asked the obvious follow-up—which of these bank properties explains success—and report a null. Correlating each column against held-out success over the twelve orderings gives $|r| \leq 0.55$ everywhere, with signs that flip between benchmarks and between writers. The analysis is exploratory, with twelve points per cell, and no main-text claim rests on it. It bears on admission rules: a policy that scores entries on summary properties of the bank would, on this evidence, have no consistent signal to optimise.

Contents versus placement. An exploratory pair of controls, run before the manifests of §2 were frozen (seed 11, held-out set pooled over three split seeds, temperature 0), asks whether order acts only through where retrieved entries sit in the prompt. *Placement* freezes a trained bank and reruns evaluation with the retrieved entries shuffled inside the prompt; the bank's hash is unchanged. *Swap* hands one ordering's bank to another run mid-stream and lets each finish its own stream. We count held-out tasks whose outcome flips, among tasks at least one arm solves. On MBPP+ placement flips $2/74$ (2.7%), at the level of batched-inference noise, while swap flips $49/263$ (18.6%), redistributing which tasks are solved with a net change of only $+0.5$ points. On HumanEval+ the two are close ($5/91$ against $19/291$), consistent with its smaller dependence on accumulated experience. With five and sixteen units per cell, this exploratory control argues against prompt placement as the main carrier on MBPP+.

Emptying the bank. If the order mattered only through a long accumulated history, then wiping the bank at fixed intervals would remove it. Figure 6 shows the opposite. Emptying every twenty rounds leaves the spread at 4.5 to 5.6 points on all four benchmarks, above the three-test-case grain everywhere, and on MBPP and LiveCodeBench it *enlarges* the spread while mean success stays within 0.7 points of the continuous bank. Order therefore acts inside every window, and a deployer who resets the store still faces the choice.

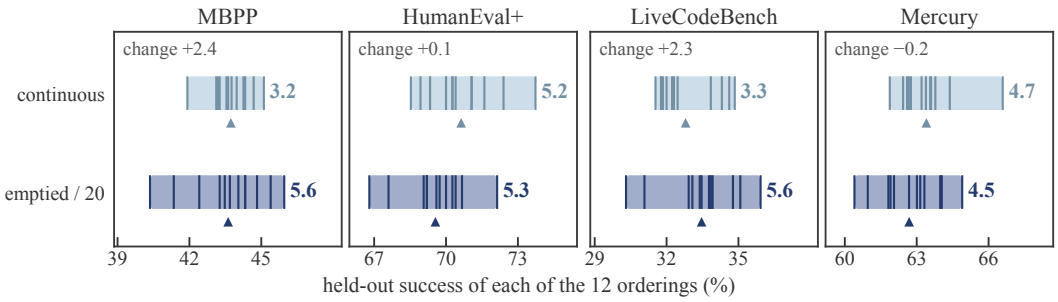


Figure 6: **Emptying the bank every twenty rounds does not remove the order effect.** Each bar spans the twelve orderings under a continuous store and under periodic reset; ticks mark single orderings, triangles the mean, and the number at the right is the spread. Ten seeds per ordering; the change in spread follows each name, with intervals $[+1.63, +3.05]$ for MBPP, $[-0.93, +0.93]$ for HumanEval+, $[+0.39, +3.15]$ for LiveCodeBench and $[-1.83, +1.33]$ for Mercury.

G THE RELIABILITY STATISTIC IN DETAIL

Table 12 gives the reliability reading of §5.3 on the four main pools, and Table 13 opens the statistic itself. The question a deployer asks is whether $\widehat{\rho}$ is a property of the pool or an accident of which seeds the pilot happened to draw, so we recompute it over every one of the 126 ways to split ten seeds into two halves of five.

Eq. 4 is computed from the pilot runs alone and needs no extra compute. With three pilot seeds the pilot has already produced twelve order means per seed. For each seed in turn we hold that seed out, correlate its twelve order means against the twelve means of the remaining two, and average the three correlations. The statistic therefore asks a single question—if we had run one fewer seed, would we have ranked the orders the same way—and answers it from runs that have already been paid for. Two properties follow and both matter for deployment. It is scale-free, so a pool on which every order scores within a point can still read high if the ranking is consistent, and a pool with a large spread can read low if the ranking moves; this is why Table 12 shows HumanEval+ with the largest Δ and only the second-highest reading. And it never touches the seeds the gain is reported on, so the reading and the payoff are statistically independent, which is what lets Figure 4(b) treat the reading as a prediction rather than as a restatement.

Table 12: **Reliability reading on the four main pools.** $\widehat{\rho}$: Eq. 4 on pilot seeds 0–2. *Confirmation*: the same correlation on seeds $5, 6$ vs. 7–9; Δ: spread of ten-seed order means; G: held-out gain on seeds 5–9. *Post hoc.

pool	pilot $\widehat{\rho}$	reading	confirmation	Δ (pt)	G (pt)	95% CI
MBPP	0.844	adopt	0.950	3.19	$+1.50$	$[+1.02, +1.99]$
HumanEval+	0.697	grey	0.662	6.13	$+2.87$	$[+2.02, +3.56]$
LiveCodeBench	0.592	grey	0.348	3.00	$+0.33$	$[-1.09, +1.76]$
Mercury*	0.501	grey	0.642	4.83	$+2.40$	$[+1.48, +3.32]$

Two readings follow. First, the statistic ranks the pools the same way however the seeds are cut: MBPP and HumanEval+ sit at the top under every split, LiveCodeBench and Mercury below them, and the ranking matches the one the two-versus-three pilot reports on untouched seeds. Second, the width of the five-versus-five range is measurement noise rather than instability in the pool. Extrapolating each cell to the full ten seeds by the Spearman–Brown relation lifts every reading to 0.75 or above, so a deployer who can afford more pilot seeds buys a sharper reading of the same underlying quantity. This is why the protocol of §5 reads $\widehat{\rho}$ as a three-way decision with a grey band rather than as a point estimate.

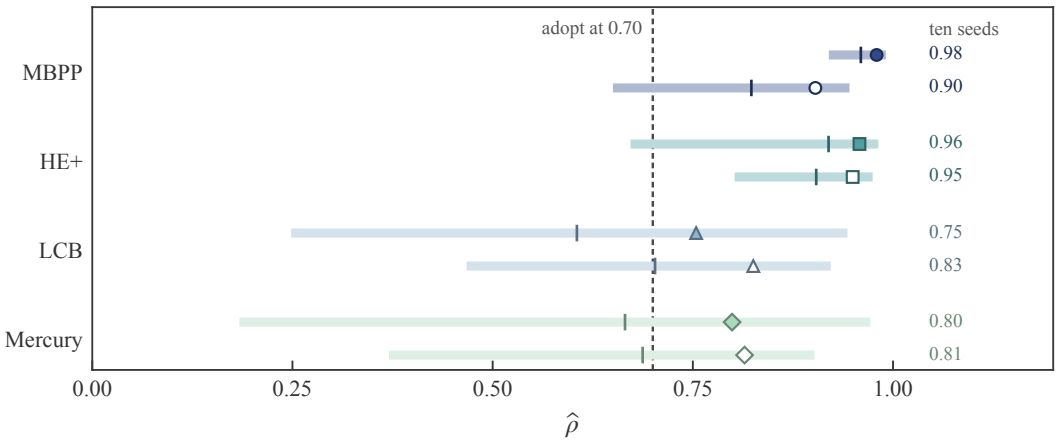


Figure 7: **The reliability statistic across every half-split of the ten seeds.** Pale bar: minimum to maximum over the 126 five-versus-five splits; tick: their mean; marker and printed value: the Spearman–Brown value at ten seeds, filled for the governed writer and open for the ungoverned bank. Dashed rule: the pre-registered adoption threshold of Eq. 4. Every ten-seed value clears it.

Table 13: **The reliability statistic under every split of the ten seeds.** *Two vs. three*: Eq. 4 on the development and on the untouched confirmation seeds. *Five vs. five*: mean, minimum and maximum over all 126 half-splits. *Ten seeds*: the five-versus-five mean carried to ten seeds by Spearman–Brown.

		two vs. three		five vs. five (126 splits)			ten
benchmark	bank	dev.	conf.	mean	min	max	seeds
MBPP	governed	0.903	0.950	0.960	0.920	0.991	0.980
MBPP	ungoverned	0.758	0.754	0.823	0.650	0.946	0.903
HumanEval+	governed	0.671	0.662	0.919	0.672	0.982	0.958
HumanEval+	ungoverned	0.751	0.884	0.904	0.802	0.975	0.950
LiveCodeBench	governed	0.474	0.348	0.605	0.248	0.943	0.754
LiveCodeBench	ungoverned	0.231	0.799	0.703	0.468	0.922	0.825
Mercury	governed	−0.225	0.642	0.665	0.184	0.972	0.799
Mercury	ungoverned	0.158	0.685	0.687	0.371	0.902	0.815

H PER-POOL RESULTS ON THE 34 FRESH POOLS

Table 14 opens every pool behind the summary of Table 5. Each pool is an independent draw: a resampled MBPP or LiveCodeBench universe, or a disjoint partition of HumanEval+. For each one the pilot runs on seeds 0–2, reports $\widehat{\rho}$, picks an order, and that order is then scored on seeds 3–4, which the pilot never saw.

The realised gain is positive on all 34 pools, from $+0.24$ to $+6.11$ points, and the three families agree on the mean to within 0.41 points despite differing in benchmark, in pool construction and in stream length. The reading behaves as a filter rather than as a predictor of magnitude: the two pools that read *decline* return $+1.26$ and $+0.56$, the four smallest gains in the table all sit at or below the adopt threshold or in its immediate neighbourhood, and the largest gains come from pools whose pilot read above 0.95.

Two cheaper versions of the protocol run on the same 34 pools. Dropping the pilot from three seeds to two cuts it to 24 trials and returns $+1.53$, $+2.48$ and $+2.07$ points on the three families, which is 84, 99 and 99% of what the full pilot recovers; the loss falls entirely on MBPP, the family whose per-pool gains are smallest to begin with. Restricting the candidate set to the ten orderings that need no reference solution—the eight random permutations, blocked-by-family and interleaved—returns $+1.23$ $[+0.89, +1.61]$, $+2.08$ $[+1.20, +3.07]$ and $+1.56$ $[+0.98, +2.26]$. An operator who cannot

rank tasks by difficulty offline therefore keeps most of the harvest, which matters because difficulty ranking is the one input in this protocol that a live request stream does not supply.

Table 14: **Every one of the 34 fresh pools gains, and the reading ranks them.** Each row is one freshly drawn pool. $\widehat{\rho}$: Eq. 4 on pilot seeds 0–2; *read*: the pre-registered three-way decision at 0.70 and 0.45; *gain*: the selected order minus the mean over the twelve, on seeds 3–4. The final row gives the column means and the count of each reading.

MBPP, resampled				HumanEval+, split				LiveCodeBench, resampled			
pool	$\widehat{\rho}$	read	gain	pool	$\widehat{\rho}$	read	gain	pool	$\widehat{\rho}$	read	gain
1	0.863	adopt	+2.54	1	0.994	adopt	+6.11	1	0.573	grey	+1.67
2	0.836	adopt	+0.24	2	0.983	adopt	+2.33	2	0.606	grey	+2.63
3	0.816	adopt	+1.93	3	0.910	adopt	+2.56	3	0.964	adopt	+0.93
4	0.909	adopt	+1.98	4	0.408	decline	+0.56	4	0.911	adopt	+1.90
5	0.989	adopt	+4.14	5	0.607	grey	+0.78	5	0.974	adopt	+5.56
6	0.954	adopt	+1.78	6	0.963	adopt	+2.00	6	0.941	adopt	+1.70
7	0.600	grey	+1.61	7	0.513	grey	+1.44	7	0.820	adopt	+1.60
8	0.756	adopt	+1.81	8	0.894	adopt	+2.11	8	0.873	adopt	+0.49
9	0.887	adopt	+1.89	9	0.934	adopt	+2.11	9	0.561	grey	+1.70
10	0.757	adopt	+0.65	10	0.687	grey	+2.56	10	0.839	adopt	+2.05
11	0.863	adopt	+2.33					11	0.972	adopt	+1.34
12	0.072	decline	+1.26					12	0.714	adopt	+0.97
mean	0.775	10/1/1	+1.85		0.789	6/3/1	+2.26		0.812	9/3/0	+1.88

I PROMPTS AND ENTRY TEMPLATES

Everything the agent reads and everything the store writes is fixed text with named slots. We reproduce all of it here, because the order effect is a statement about what fills those slots and when.

System message. One string, identical for every arm, every ordering and every seed:

```
You are a careful Python programmer.  Implement exactly the
requested function under the name `solution`.  Reply with
ONE
"`python fenced code block containing `def solution(...)`
and nothing else.  No input(), no file or network access;
standard library only.
```

Task message. One string per round, assembled from up to four blocks. The third block is the only one that depends on the history of the run, and therefore the only channel through which arrival order can reach the generator:

```
Task:  {task statement}
The function takes these arguments, in order:  {arguments}.
General strategy learned so far:
{workflow, when the arm maintains one}
Related experience retrieved from your own memory (may or
may not be relevant):
{retrieved entries}
Now write `solution`.
```

Entry templates. An entry is produced by a template, not by a second model call. This is deliberate: an LLM-written entry would give the writing arm extra inference compute and confound the comparison of Table 7, where every arm of the 2×2 makes an identical number of generator calls. The three levels are

```
[memory/{family}] task {id}:  {task statement}
verified solution ({verdict}):
"`python {code} "`
[skill/{family}] {id}:
```

```
"'python {code} "'
[rule/{family}] to {goal} -> use {constructs}.
```

where `{goal}` is the task statement truncated at ninety characters after its leading clause and `{constructs}` is the sorted set of algorithmic constructs detected in the accepted code, or `direct expression` when none is. A rule therefore keeps the family, the goal and the algorithmic shape and drops the code, which is what makes it the lossy level of Table 11. The router picks the level from how often the family has been seen: a family seen at least four times routes to `rule`, a family seen at most once routes to `memory`, and everything between routes to `skill`. Because that count is a running count, the same solved task yields a different level under a different arrival order.

The ported writer. The ReasoningBank arm of Table 9 replaces all of the above with the official module's own text. Its two distillation instructions, its at-most-three-items output contract—`# Memory Item i / ## Title / ## Description / ## Content`—and its retrieval instruction are carried over byte for byte from the released implementation, including the sentence that prefixes every retrieval query:

```
Given the prior software engineering queries, your task is
to analyze a current query's intent and select relevant
prior queries that could help resolve it.
```

That arm writes on failures as well as successes and applies no admission filter and no capacity bound, which is why its bank ends $9.33\times$ larger than ours, and it still carries a 4.86-point spread.

J Reproducibility details

Units and streams. One *round* is one training task; a *trial* or *run* is one pass of the agent over a pool's full training stream under one ordering and seed, followed by held-out scoring with the bank frozen. Every trial starts from an empty bank. Suite manifests freeze, per benchmark, the task universe, the train/held-out draw (split seed), the eight random permutations (order seed) and the four constructed orders; the twelve names are `random_0`–`random_7`, `easy_first`, `hard_first`, `blocked_by_family` and `interleaved`. The LiveCodeBench stream has 110 rounds.

Pre-registration. Each confirmatory contrast was written down with its estimator and falsification threshold before its results were read: the fresh-pool feasibility test on MBPP and Mercury (2026-09-18, 15:00, before any fresh-pool result existed); the head-to-head against the three literature writers (2026-09-19, 01:40, registered twelve minutes after launch and before any success value was read); the HumanEval+ fresh-pool test, which fixes the $0.70/0.45$ thresholds of Eq. 4 with a Pearson estimator (2026-09-20); and the backbone sweep, doubled pool, seed extension and ReasoningBank port (2026-09-21). Table 15 puts the thresholds beside what the runs returned. Mercury was excluded from confirmatory claims under the first document, because its development-seed ranks were unstable (Pearson -0.23); it is reported descriptively, and its post hoc pilot gain ($+2.40$, Table 12) is not counted as confirmatory evidence. The LiveCodeBench fresh pools were run after these documents and use their estimators unchanged.

Statistics. Intervals on a single pool resample seeds with replacement, keeping all orders and arms of a seed together so that contrasts stay paired, $10{,}000$ replicates. The pilot's pick is fixed by seeds disjoint from those it is scored on, so selection sits outside the resampling and adds no optimism to the interval. Fresh-pool intervals resample pools. Holm corrections run over the benchmarks within each pre-registered family. The spread's permutation test shuffles order labels within each seed ($2{,}000$ draws). Fresh pools share part of their training sets: training sets overlap by about 25% between MBPP pools and by 0.68 on average (0.75 at most) between HumanEval+ partitions, so their intervals describe the protocol over this family of pools rather than over all possible pools.

Writers. The three literature writers reproduce each method's entry type and write trigger inside our store; retrieval, prompting and control flow are ours for every writer, so they are write policies under a common harness rather than three separate systems. Reflexion writes one verbal reflection

Table 15: **Thresholds fixed before the runs, beside what the runs returned.** Covers the registrations that extend the main-text claims beyond the published grid; each threshold is quoted from the document named in its row, and each document was frozen before any result file of that experiment existed.

document (frozen)	quantity	threshold set in advance	returned
backbone sweep (2026-09-21 13:45)	spread, 7B on MBPP	≥ 2.0 pt	5.82
	spread, 32B on MBPP	≥ 2.0 pt	5.53
	spread, 72B on MBPP	≥ 2.0 pt	4.68
doubled pool (2026-09-21 17:25)	spread, 480 tasks	≥ 2.0 pt	5.19
	spread vs. original	$\leq 1.5 \times 3.69$ pt	5.19
	pilot gain	> 0	$+2.19$
seed extension (2026-09-21 17:25)	HumanEval+ gain, four writers, seeds 5–14	significant in $4/4$ cells	$4/4$
	sign agreement across the two seed blocks	≥ 10 of 12 cells	$12/12$

after each failed task and stores it without a gate, so its bank grows with the failures a given order happens to produce. ExpeL extracts one distilled insight every twenty tasks from the recent log, so its writes are paced by the clock rather than by the outcome. AWM induces one workflow every five solved tasks, so its writes are paced by success. The three therefore differ in what triggers a write as well as in what a write contains, and the order of arrival changes all three triggers: which tasks fail, what the recent log holds at each twentieth task, and when the fifth success occurs. This is why the copied-order result of §5 is a statement about writers and not only about pools. Per-unit results, analysis scripts and the pre-registration documents are in the supplementary material.