DECISION-SUFFICIENT STATE REPRESENTATIONS: MEASURING AND REDUCING WRITE-TIME REGRET

Anonymous authors
Paper under double-blind review

ABSTRACT

Long tasks produce more history than an LLM agent can hold in its context, and more than it uses reliably even when the history fits. A growing line of work therefore has agents carry a short written state instead: at every step a writer rewrites the state, and a reader acts from the state alone. Steps stay cheap, but anything the writer drops is lost before later decisions reveal that they need it. We quantify this loss and ask whether training can reduce it. Comparing the written state with the best state of the same size written in hindsight, we split the reader's loss into a budget loss, which any state of that size must incur, and a write-time regret, which comes from the writer's choices. In TextWorld cooking games where we control how long a fact must be carried before it is needed, a 128-token state holding the facts wins nearly every game, while prompted language-model writers win at most 17 %. Almost all of the loss is write-time regret, and it grows with the delay. Creating a real delay takes care: steps inserted between a fact and its use delay it only if they do not show the fact again. We then train the writer from the reader's own loss. DSSR (decision-sufficient state representations) scores candidate states by how well the reader acts after the writer carries them forward, and teaches the writer to prefer the better ones. This forward-rolled score predicts game outcomes ($\rho = 0.48$; 0.26 under the delayed protocol), whereas scoring a candidate as a fixed context, as hindsight methods usually do, does not ($\rho \leq 0.07$). On a pre-registered test split opened once, training adds $+7.0$ $[+1.9, +12.2]$ points of success when facts are needed soon, bringing a plain summary writer to the level of belief- and slot-based memory prompts. The gain shrinks as the delay grows: on held-out games it is only at the shortest delay, and our pre-registered tests for a gain under long delay fail. A hand-written rule that names what to keep, using task knowledge the writer is not given, does better at every delay. We trace this limit to credit assignment: keeping a fact now pays off only if every later rewrite keeps it too, which a per-step score cannot see. Anonymous codebase: `https://anonymous.4open.science/r/AgentDSSR`.

1 INTRODUCTION

LLM agents outgrow their context windows: a household task, a web session or a text game can run for hundreds of steps and many thousands of tokens of observations. A widely used remedy bounds the agent's working memory (Wang et al., 2023; Packer et al., 2023; Zhou et al., 2025; Singh et al., 2026). A writer condenses the previous memory and each new observation into a fixed-size text state, and a reader selects actions from that state and the current observation, so every step costs the same however long the episode has run. The price is irreversibility. Raw observations are dropped after each rewrite, and a fact reaches the step that needs it only if every intermediate rewrite has preserved it (Figure 1a). Prompt compression and retrieval-based memory avoid this by keeping the raw record and compressing it once the query is known (Figure 1b). A constant-context agent must commit at write time; this paper measures what that commitment costs and asks whether the writer can be trained to make it well.

Two things can go wrong: the state can be too small to hold the facts that matter, or the writer can fill it with the wrong facts. Benchmarks that report only task success (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2026) cannot tell these apart, yet only the second can be fixed by a better writer. We separate them (§3) by comparing the reader's decisions under three inputs: the full history, the

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.

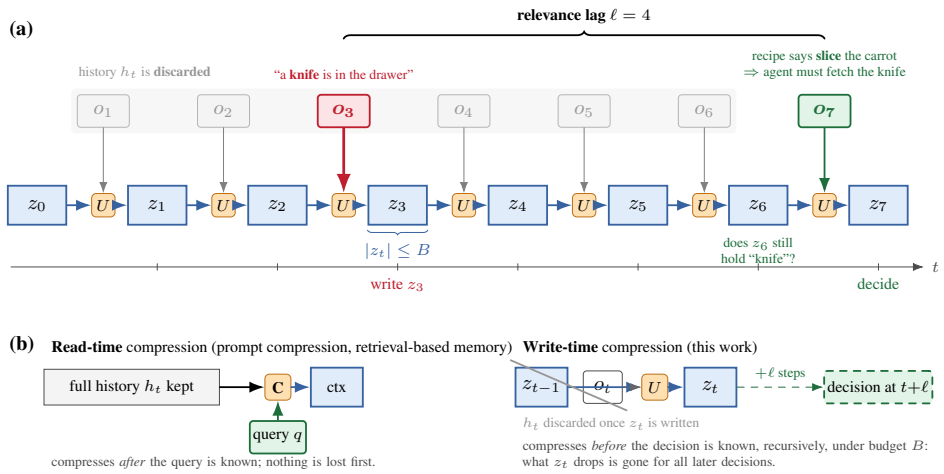

Figure 1: Write-time compression under relevance lag. (a) One episode of a constant-context agent: at every step the writer U rewrites the state z_t ($\le B$ tokens) from z_{t-1} and the new observation o_t, and the history h_t is discarded. A fact observed at step 3 (the knife's location) is needed at step 7 (the recipe says *slice*), $\ell = 4$ rewrites later; whether z_6 still holds it was decided by U before the need was known. (b) Read-time compression (prompt compression, retrieval-based memory) keeps the history and compresses once the query is known; write-time compression (this work) compresses before the decision is known, recursively and under budget: what z_t drops is gone for every later decision.

best state of the same size that could have been written in hindsight, and the state the writer actually wrote. The first difference is the *budget loss*, what any state of that size must give up; the second is the *write-time regret*, what the writer gave up by choosing without knowing the future.

We measure both in TextWorld cooking games (Côté et al., 2018; Adhikari et al., 2020), where we control how long a fact must be carried before it is needed. The budget is not the problem: a 128-token state holding the true facts wins nearly every game, while every state written by a language model under five prompt designs modelled on published memory systems wins at most 17 %. Almost all of the loss is write-time regret, and it grows the longer a fact must survive repeated rewriting. Controlling this delay turned out to be subtle: steps inserted between a fact and its use create a real delay only if they do not show the fact again (§5.2).

We then train the writer with DSSR (decision-sufficient state representations; §4). At a write step the writer proposes several candidate states, the frozen reader scores each by how well it makes the next few decisions from it, and the writer learns to prefer the better candidates. What matters is how a candidate is scored: the reader must see the state as the writer will actually carry it forward, not the candidate followed by the raw steps, as hindsight methods usually assume. Only the first form predicts how the game ends ($\rho = 0.48$ against 0.07), because a state's value depends on what the writer will later drop from it.

On a test split opened once, after every choice was fixed, the trained writer wins $+7.0$ [+1.9, +12.2] points more games than the same writer without training when facts are needed soon after they are seen, to the level of the belief- and slot-based prompts (§5.4). The gain shrinks as the delay grows and is significant on held-out games only when facts are needed soon; our pre-registered tests for a gain under long delay fail. A hand-written rule that tells the writer which kinds of facts to keep helps at every delay, but it relies on knowledge of the task that the trained writer does not receive. We trace the limit to credit assignment (§5.6): keeping a fact now pays off only if every later rewrite keeps it too, and a per-step score cannot see that.

Contributions: (i) A decomposition of a constant-context agent's decision loss into budget loss and *write-time regret*, a protocol verified to create delay, and the finding that the writer's choices, not the budget, are the bottleneck. (ii) A *recursive sufficiency score* that evaluates a candidate state by rolling the writer forward through it, and evidence that only this form predicts outcomes. (iii) DSSR, which trains the writer from this score, with a significant gain on a single-use test split when facts are

needed soon and a gain that shrinks with delay. (iv) A diagnosis of why a per-step objective cannot teach long-range retention.

2 Related work

Memory and state for LLM agents: Recursive summarisation rewrites a running summary as the interaction proceeds (Wang et al., 2023); MemGPT pages information between context and external storage (Packer et al., 2023); Reflexion, generative agents, MemoryBank and A-MEM keep reflections and retrievable memories (Shinn et al., 2023; Park et al., 2023; Zhong et al., 2024; Xu et al., 2025). Closer to our writer are compact-state designs: belief claims with verbalised uncertainty (Singh et al., 2026), progress-aware retention (Jiang et al., 2026), competing conclusions with probabilities (Liao et al., 2026), optimised compression guidelines (Kang et al., 2025), context folding (Sun et al., 2025), and MEM1's end-to-end reinforcement learning of a constant-memory agent (Zhou et al., 2025). They propose memory designs; we measure where a fixed design loses and train the writer alone from a frozen reader's loss. Our prompted formats (§5.1) mirror this literature: a free-text summary, a belief list with certainty tags (Agent-BRACE-like), fixed slots (PABU-like), and a summary with a hand-written retention guideline of the kind ACON optimises.

Context compression and evaluation: Gist tokens, in-context autoencoders, AutoCompressors and LLMLingua compress a *given* context at read time (Mu et al., 2023; Ge et al., 2024; Chevalier et al., 2023; Jiang et al., 2023); our read-time oracle z^{rt} is such a compression chosen with hindsight, and write-time regret measures how far a prospective writer falls short of it. Long contexts do not remove the problem, since models use them unevenly (Liu et al., 2024). LoCoMo, LongMemEval and MemoryAgentBench measure the end accuracy of memory systems over long interactions (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2026); our protocol adds a decomposition of the loss into budget and writer terms and a lag manipulation verified to create lag.

Hindsight and preference training: Hindsight relabeling turns achieved outcomes into supervision (Andrychowicz et al., 2017; Zhang et al., 2023). DSSR relabels a write with the reader's future loss and trains the writer with direct preference optimization (Rafailov et al., 2023), adding a likelihood term on the chosen sample as in iterative reasoning preference optimization (Pang et al., 2024). The readers are behavior-cloned with DAgger iterations (Ross et al., 2011).

3 Constant-context agents and write-time regret

The constant-context agent: A writer U maintains a text state z_t of at most B tokens, updated recursively from the goal g, the previous state, the last action, and the new observation, $z_t = U(g, z_{t-1}, a_{t-1}, o_t)$; the raw history is discarded after each write. A frozen reader π chooses the action from (g, z_t, o_t) alone. Nothing but z_t carries information across steps: it plays the role of a belief state (Kaelbling et al., 1998), and the design has constant context by construction. We study the writer: the reader is trained once, on states from untrained writers, and never changes.

Decision loss and its decomposition: Consider a decision point t with reference action a_t^* (the first command of the environment's optimal policy) and full history h_t. For any context c let $\mathrm{NLL}_t(c) = -\log p_\pi(a_t^* \mid g, c, o_t)$, with p_π the reader's distribution over the admissible options. The *sufficiency gap* of a state format M is $\delta_t(M) = \mathrm{NLL}_t(z_t^M) - \mathrm{NLL}_t(h_t)$. To separate budget from writer we define a *read-time oracle* state z_t^{rt}: the untrained writer compresses the entire history h_t in one shot at budget B, $N = 8$ samples are drawn, and the one with the lowest NLL_t is kept. z_t^{rt} is a hindsight construction, since it sees the whole history and is selected with a_t^*; it is a hindsight reference for what B tokens written by this writer can carry for this decision, not a bound: a recursively written state can do better at a given decision. Then

$$\delta_t = \underbrace{\mathrm{NLL}_t(z_t^{\mathrm{rt}}) - \mathrm{NLL}_t(h_t)}_{\beta_t \text{ (budget loss)}} + \underbrace{\mathrm{NLL}_t(z_t^M) - \mathrm{NLL}_t(z_t^{\mathrm{rt}})}_{\kappa_t(M) \text{ (write-time regret)}}. \tag{1}$$

β is what a B-token state must lose even with hindsight; κ is what the recursive writer lost on top of that by choosing, prospectively, what to keep.

Relevance lag: To relate regret to time we measure, for each decision point, which past step matters: the *relevance lag* $\ell_t = t - s^*$, where s^* is the chunk of h_t whose removal increases $\mathrm{NLL}_t(h_t)$ most

(chunks are defined in Appendix D), undefined when no removal costs at least 0.5 nats; ℓ is measured on the actual history, whatever the protocol intended.

Controlled lag protocol: Episodes have two phases. In Phase A a scripted explorer visits every room, opening every container and door, so that every fact a later decision needs (ingredient locations, the knife, the appliances, the map) has been observed; it then walks to the kitchen and reads the cookbook, which reveals the recipe. In Phase B the reader acts until the game is won or lost. The writer updates z_t at every step of both phases. Lag is manipulated by a seeded random walk of $F' \in \{0, 10, 20, 40\}$ steps among visited rooms, placed *after* the reveal (protocol F'); the walks for different F' are nested prefixes of one walk. Our pre-registered design placed the walk *before* the reveal and did not create lag (§5.2); protocol F' was adopted before the writer was trained and before the test split was opened (original-protocol results in Appendix F).

4 DSSR: TRAINING THE WRITER FROM THE READER'S LOSS

From episodes to single writes: Training the writer on episode outcomes would mean multi-turn reinforcement learning through a frozen reader with a sparse, delayed reward. DSSR instead decomposes the writer's problem into single-turn write problems. At a write point t, sibling candidate states $z^{(1)}, \ldots, z^{(n)}$ sampled from the current writer are compared by how well the frozen reader can make the *future* decisions from them, and the writer is trained to prefer the better candidates. The signal is the quantity we want to reduce, the reader's loss.

Hindsight sufficiency score: For write point t and candidate z, the targets are the decision steps u of Phase B with $t < u \leq t + 40$, evenly subsampled to at most K (default $K = 12$). The score is the mean reader log-likelihood of the reference actions at the targets,

$$S_K(z) = \frac{1}{|T_K|} \sum_{u \in T_K} \log p_\pi\big(a^*_u \mid g,\ c_u(z),\ o_u\big), \tag{2}$$

where $c_u(z)$ is the context the reader sees at u. Two forms of c_u are possible. The *window* form is z followed by the raw steps $t{+}1, \ldots, u{-}1$: it credits z only for information from before t and is cheap, and it is the form most hindsight schemes would use. The *recursive* form rolls the writer forward from z through the logged steps, $z_k = U(g, z_{k-1}, a_{k-1}, o_k)$ at temperature 0, and the reader sees only z_u, exactly as it would online. The two forms disagree completely for a lossy writer (§5.2): the value of a state depends on how it will be rewritten, and the window form hides this. DSSR uses the recursive form.

Optimization: Each round collects episodes on the training games with the current writer and the frozen reader, samples write points from both phases, writes n candidates per point at temperature 1, scores them with S_K, and trains. Round 0 is supervised fine-tuning on the best candidate by S_{12} (best-of-n distillation, as in reward-ranked fine-tuning, Dong et al., 2023). Rounds 1–3 are direct preference optimization (Rafailov et al., 2023) on best-versus-worst pairs whose score gap is at least 0.3 nats (at most two pairs per write point), with $\beta_{\text{DPO}} = 0.1$ and a 0.5-weighted likelihood term on the chosen state (Pang et al., 2024); the previous round's adapter is both the initialisation and the reference. The writer is a LoRA adapter (Hu et al., 2022); the prompt is the unchanged summary prompt, so DSSR and the SUMMARY baseline differ only in the adapter. Model selection uses online success on the validation games, and the selected configuration is retrained with three seeds. Variants trained from the same candidates with the same recipe isolate the objective: horizon $K \in \{1, 4, 12\}$; NO-HINT, which rejects candidates that contain an admissible command verbatim; and OUTCOME, which scores a candidate by the true return of continuing the episode from it, the outcome-reward alternative.

Reachability-restricted sampling: Section 5.6 shows that under lag most write points offer no candidate that still contains the decisive fact. The *A4* variant keeps a post-reveal write point only if its input state still holds at least one recipe directive, so that keeping versus dropping is a choice the candidates can make, and writes 12 candidates at temperature 1.1 so that both choices appear. Everything else is unchanged. This variant produced the selected writer.

Table 1: Validation games under protocol F', $B = 128$: online success (% of 45 games won) and mean write-time regret κ (nats) per lag; success at $B = 256$ for comparison. FULL has no budget; ORACLE-B is privileged (≈ 47 tokens). κ is the mean of per-game means; β lies between -0.14 and 0.23 in every B=128 cell. Full grid with δ, β, κ and intervals in Appendix F.

	success, B=128				κ, B=128				success, B=256			
F'	0	10	20	40	0	10	20	40	0	10	20	40
FULL	97.8	97.8	97.8	97.8	–	–	–	–	–	–	–	–
ORACLE-B	100	100	100	97.8	−0.19	−0.17	−0.21	−0.15	100	100	100	97.8
LASTK	0	0	0	0	0.71	1.80	1.46	1.46	2.2	0	0	0
SUMMARY	2.2	2.2	0	2.2	0.85	1.26	1.47	1.10	20.0	6.7	2.2	2.2
SLOTS	11.1	0	0	0	0.49	1.55	1.38	1.69	15.6	4.4	2.2	0
BELIEF	8.9	6.7	4.4	6.7	0.79	1.44	1.27	1.17	28.9	15.6	8.9	6.7
GUIDE	11.1	11.1	2.2	6.7	0.66	0.72	0.84	1.10	40.0	31.1	31.1	22.2

5 EXPERIMENTS

5.1 SETUP

Environment and splits: We use TextWorld cooking games (Côté et al., 2018; Adolphs & Hofmann, 2020; Adhikari et al., 2020) with three-ingredient recipes, cutting and cooking, and 6, 9 or 12 rooms (240 train, 45 validation and 90 test games, disjoint seeds; the validation and test games contain unseen foods and preparations). A game is won by preparing and eating the meal and is lost, irreversibly, on the first wrong cut or cooking verb. Full histories run from 2.2k tokens (6 rooms, $F' = 0$) to 9.1k on average (12 rooms, $F' = 40$), against state budgets $B \in \{64, 128, 256\}$. The reader chooses among the admissible commands, presented as lettered options and shuffled with a fixed seed; the reference action a^* is the first command of the game's optimal policy, which is privileged and never shown to the writer or reader (a string search over all logged prompts enforces this). Details and environment statistics are in Appendix A.

Models and formats: Writer and reader R_1 are Qwen3-4B-Instruct (Yang et al., 2025) with separate LoRA adapters served by vLLM (Kwon et al., 2023) with batch-invariant kernels, so every number is bit-reproducible on one GPU. R_1 is behaviour-cloned on the reference action over 19 context variants (the full history and each prompted format at each budget, from *untrained* writers), with two DAgger iterations (Ross et al., 2011); it wins 97.8 % of validation games with the full history and is frozen. A second reader R_2 (Phi-4-mini, Abouelenin et al., 2025, same recipe; 77.8 % with the full history) tests whether trained states are informative to a reader they were not tuned to. The prompted formats are SUMMARY, BELIEF, SLOTS, GUIDE (the summary prompt plus a hand-written rule: always keep food, tool and appliance locations, open containers and doors, and the room graph) and LASTK (the most recent raw steps that fit), as in §2; ORACLE-B is a privileged state recomputed from the game's true facts and cut to B tokens, which shows what the budget can carry; FULL is the unbounded history. The prompts are reproduced in Appendix C; DSSR uses the SUMMARY prompt.

Protocol, statistics, pre-registration: Every cell is 45 (validation) or 90 (test) deterministic episodes per lag $F' \in \{0, 10, 20, 40\}$; κ is audited on up to six decision points per episode and every κ reported per (format, lag) cell is the mean of per-game means. Confidence intervals are paired percentile bootstraps over games (10,000 resamples), and all numbers in a table come from a single GPU. Hypotheses, gates and the analysis plan were written before the corresponding data existed; two amendments made after the diagnostic (the recursive score and protocol F') were recorded before the writer was trained, and the test split was opened once, after selection and seeds were fixed on validation (Appendix E).

5.2 THE BOTTLENECK IS WRITE-TIME REGRET, AND IT GROWS WITH LAG

The budget suffices; the writer does not choose well: Table 1 gives the validation grid (drawn in Appendix F, Figure 6). With the full history the reader wins 97.8 % of the games at every lag; the privileged ORACLE-B state, about 47 tokens, wins 98–100 % at every budget. Every LLM-written

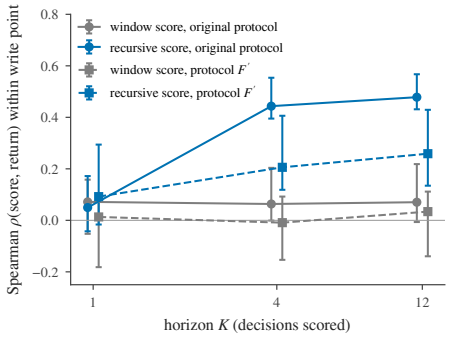

Figure 2: The sufficiency score predicts return only in its recursive form: mean within-write-point Spearman correlation between a candidate state's score S_K and the realised return of the episode continued from it (200 write points $\times$ 8 candidates from SUMMARY/B=128 trajectories, continued to the end of the game; game-clustered 95 % intervals), for the window and the recursive score at $K \in \{1, 4, 12\}$ under both protocols.

state at $B = 128$ wins 0–11 %, at $B = 64$ none of them wins anything, and at $B = 256$ the best, GUIDE, falls from 40 % to 22 % as F' grows. Decomposing the gap on 200–270 decision points per cell, κ is 0.4–2.0 nats in every budgeted LLM cell (intervals exclude zero) while $|\beta| \leq 0.40$; ORACLE-B has $\kappa < 0$: a state written with knowledge of what matters beats the best of eight hindsight compressions of the whole history. Across the 76 (format, budget, lag) cells the mean gap δ predicts online success with Spearman $\rho = -0.84$ (-0.90 on the replicate GPU), so the offline loss is a valid proxy for the online outcome.

Regret grows with rewrites since the relevant observation: Under protocol F' the measured relevance lag ℓ rises from 4 to 16, 30 and 48 steps at $F' = 0/10/20/40$, and κ at $B = 128$ rises from 0.5–0.9 nats to 1.1–1.8 for the four write-time formats (GUIDE: $0.66 \rightarrow 1.10$): a large step within the first ten rewrites and a plateau after it, with β flat. Against the measured lag, κ rises from 0.61 $[0.38, 0.84]$ at $\ell \in [1, 4]$ to 1.87–2.30 at $\ell \geq 10$ (Table 7), while the full-history loss stays ≤ 0.7 and $\beta \approx 0$ in every bin, so the rise is writer loss, not task difficulty; $\kappa(\ell \geq 5) - \kappa(\ell < 5) = +1.08$ $[0.24, 1.97]$.

What counts as lag: Table 8 in Appendix F isolates the protocol result. The 40-step walk placed before the reveal leaves κ unchanged (-0.20, -0.34 and $+0.24$ nats for three formats). Suppressing the re-printed room contents during that walk raises κ at $F = 40$ by $+0.47$ $[0.10, 0.93]$: the walk had been refreshing the writer. Placing the walk after the reveal raises κ by $+1.07$ $[0.42, 1.75]$, while FULL and ORACLE-B stay at 98–100 %. Distractor steps are lag for a recursive writer only when they neither re-expose the relevant facts nor precede their revelation.

The sufficiency score predicts return only in its recursive form: We continued 200 write points $\times$ 8 candidate states from SUMMARY/B=128 trajectories to the end of the game (1,600 episodes) and correlated each score with the realised return within a write point (Figure 2). The window score is uncorrelated with return at every horizon ($\rho \leq 0.07$, both protocols) and uncorrelated with the recursive score ($\rho \leq 0.09$). The recursive score reaches $\rho = 0.48$ $[0.39, 0.56]$ at $K = 12$ (0.44 at $K = 4$, 0.05 at $K = 1$) and 0.26 $[0.13, 0.43]$ under F', where 91 of the 200 write points have identical returns for all candidates. Continuing from the logged state reproduces the logged episode exactly (200/200), so the machinery is sound: the fixed-state approximation is invalid for a lossy writer because a state's value depends on how it will be rewritten.

5.3 TRAINING AND SELECTION

Distillation: Each round collects 480 episodes on the training games (writer at temperature 0.7, reader ϵ-greedy, F' and B drawn uniformly), samples 3,984 write points (half from Phase A), writes six candidates per point and scores them recursively. Supervised fine-tuning on the best candidate lifts validation success at $F' = 0$ from 2.2 % to 13.3 %, leaves $F' \geq 10$ unchanged, and leaves κ unchanged at every lag (mean over F' 1.17 before, 1.20 after). The horizon matters already here:

Table 2: Test split (90 games), $B = 128$, reader R_1: success (%) and κ (nats, mean of per-game means) per lag F'. Trained writers are the best validation round of each variant; DSSR is the selected A4 round-2 configuration, mean of three seeds (seeds at $F' = 0$: 3.3 / 15.6 / 12.2). Intervals, R_2 rows and the budget frontier: Appendix H.

	success (%)				κ (nats)			
F'	0	10	20	40	0	10	20	40
FULL (no budget)	95.6	96.7	96.7	94.4	–	–	–	–
ORACLE-B (privileged)	98.9	100	98.9	97.8	–	–	–	–
LASTK	0	0	0	0	–	–	–	–
SUMMARY	3.3	2.2	1.1	0	0.79	1.05	1.35	1.66
BELIEF	11.1	0	0	0	0.57	1.21	1.55	1.73
SLOTS	10.0	0	0	0	0.85	1.75	1.64	1.96
GUIDE (hand-written rule)	**14.4**	**6.7**	0	2.2	**0.46**	**0.86**	1.03	**1.18**
DSSR (A4 r2, 3 seeds)	10.4	3.7	2.2	1.9	0.62	1.20	1.36	1.58
DSSR main line (r1)	7.8	1.1	1.1	1.1	0.51	1.15	1.44	1.92
OUTCOME (r3)	8.9	0	1.1	0	0.57	1.39	1.52	1.84
K=1 (r1)	6.7	0	1.1	0	0.87	1.23	1.36	1.84
K=4 (r3)	11.1	1.1	2.2	1.1	0.54	1.08	**0.96**	1.47
NO-HINT (r3)	8.9	0	**3.3**	**2.2**	0.87	1.22	1.21	1.40

distilling the best candidate by S_1, S_4 or S_{12} gives 4.4, 6.7 and 13.3 %. OUTCOME's round-0 target is degenerate: for 76 % of its write points all six candidates have the same return.

Preference rounds and selection: Round 1 (4,460 pairs) takes κ at $F' = 0$ from 0.85 to 0.52 nats and raises it at $F' \geq 10$ (1.26/1.47/1.10 $\rightarrow$ 1.60/1.59/1.87); every variant behaves the same way, for a reason §5.6 isolates on the training candidates. Policy iteration helps slowly: round 3 of the main line and of K=4 flatten the κ-versus-F' curve (lag $F' \in \{20, 40\}$: -16 % [-34, +7] and -21 % [-40, +2]), and no variant reduces lag κ significantly on 45 games (Appendix G). The A4 variant, which samples candidates only where retention is still a choice (§4), is the strongest: after two rounds it wins 26.7 / 11.1 / 4.4 / 2.2 % at $F' = 0/10/20/40$ (untrained 2.2 / 2.2 / 0 / 2.2), and its pair signal does not thin as the policy sharpens (80 % of write points with a qualifying pair, median gap 0.88 nats, against 54 % and 0.36 for the main line at round 3). Selection on validation success picks the A4 round-2 writer (11.1 % averaged over F') ahead of K=1 round 1 (7.2 %), K=4 round 3 (6.7 %) and the main line's round 1 (5.0 %); its two additional seeds win 8.9 and 22.2 % at $F' = 0$ and reduce zero-lag κ by 28–64 % (Appendix G).

5.4 MAIN COMPARISON ON THE TEST SPLIT:

Result: Table 2 is the single-use evaluation. At zero lag the trained writer wins 10.4 % of the test games against 3.3 % for the prompted SUMMARY writer, $+7.0$ [+1.9, +12.2] points; every trained variant lands in the same band (6.7–11.1 %) and two of the three seeds are above the baseline (15.6 and 12.2 %) and the selected seed ties it (3.3 %). The comparison isolates training, since DSSR and SUMMARY differ only in the adapter; the BELIEF and SLOTS formats reach the same band at zero lag without training (11.1 and 10.0 %). Under R_2 the zero-lag ordering is reproduced (GUIDE 11.1 > DSSR 5.6 > SUMMARY 2.2 %), so the trained states are informative to a reader they were not tuned to. Under lag the trained writer is not distinguishable from the prompted writers: at $F' \in \{20, 40\}$ it is $+1.5$ [-0.4, +3.3] points over SUMMARY, $+1.5$ [+0.2, +3.0] over OUTCOME and $+0.9$ [-0.9, +2.8] over GUIDE, the best prompted state there; its lag κ ($F' \in \{20, 40\}$) is 1.47 against 1.50 untrained (-2 % [-16, +16]). Every pre-registered gate on the lag gain fails (Appendix E). The untrained writer's κ rises with F' on test (0.79 $\rightarrow$ 1.66), reproducing the validation phenomenon, and GUIDE has the lowest κ of the untrained formats at every lag and the highest success averaged over lags (at $B = 256$: 30.0 / 16.7 / 15.6 / 15.6 %; Appendix H).

Where the reduction lives: Binning the audited decision points by the number of rewrites since the reveal (Figure 3) gives the continuous form of the lag grid and the statement of the method's range. On validation the three seeds reduce κ by 81 % within 1–2 rewrites (paired difference $+0.83$ nats [+0.20, +1.65]) and by 52 % within 3–5 ([+0.10, +1.25]), by 44 % at 6–9 (not significant), by about

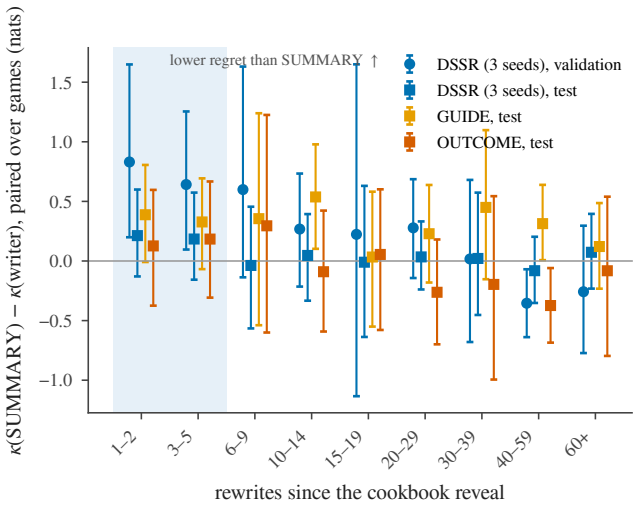

Figure 3: Where the reduction lives: paired difference of per-game mean κ between the untrained SUMMARY writer and a trained or rule-based writer (positive = lower regret than SUMMARY), per bin of rewrites since the cookbook reveal, with 95 % game-bootstrap intervals; the three DSSR seeds are pooled per game. Shaded: bins where the trained writer's reduction is significant on validation (1–5 rewrites). Per-bin curves: Figures 11 and 12.

20 % at 10–29, and not at all beyond, becoming *worse* than the untrained writer at 40–59 rewrites; on test the profile is directional only (-31 % at 1–2, -19 % at 3–5, zero from 6 on). GUIDE reduces κ by 20–50 % in every bin up to 40–59 rewrites on test. The learned writer thus helps at short (1–2 rewrites) and medium (3–5) lag, weakly at 6–9, and not at 10 or more; a rule that names what to keep helps everywhere. An exploratory validation grid with eleven filler lengths agrees (Appendix I; post hoc): the seeds' game-paired success gain over SUMMARY is $+17$ points at $F' = 0$, $+7$ to $+10$ at $F' = 2$–9 (intervals above zero), $+5$ to $+7$ at $F' = 15$–20 and within noise at $F' = 30$–40.

5.5 Ablations

The variants in Table 2 were trained from the same candidates with the same recipe (intervals in Appendix H). Measured sufficiency beats the outcome reward on the same write problems by a small but significant margin under lag ($+1.5$ [+0.2, +3.0] points), and OUTCOME's κ on test is *higher* than the untrained writer's beyond ten rewrites. $K{=}4$ has the lowest lag κ of the trained writers and $K{=}1$, the main line and OUTCOMES the highest, but no pairwise difference is significant on 90 games. NO-HINT keeps the zero-lag gain with a command-phrase fraction of 0.13–0.18 against 0.30–0.37, so the gain is not a memorised action list. At $B = 64$ no LLM writer wins more than 2.2 %; at $B = 256$ the trained writers' zero-lag gain widens but GUIDE still dominates. Round 0 supplies most of the zero-lag gain, and the test seed spread at zero lag (3.3–15.6 %) is as large as the gain.

5.6 Why the gain shrinks with delay: credit assignment across rewrites

The reduction of regret is largest within a few rewrites of the reveal and shrinks after (Figure 3). Two measurements on the round-0 training candidates explain this profile (2,818 post-reveal write points; coverage is the fraction of the recipe's verb–ingredient directives present in a state; Figure 4 and Appendix G). First, candidates inherit their input. The six candidates are rewrites of the same previous state, and their mean coverage (0.24) equals its coverage (0.24); at 51 % of post-reveal write points (67 % at $F' = 40$) no candidate contains any directive, so once a rewrite has dropped the recipe, best-of-n cannot select what is absent. Second, the score is taken under a forgetful continuation. Where candidates do differ in coverage (9 % of points), S_{12} prefers the more complete one only at chance level (28 % against 29 %): the candidate is handed to the current writer, which drops the recipe within a few rewrites, while the decisions that would reward keeping it are 10–40 steps away. Both effects grow with delay, and the training signal thins accordingly: the median best-minus-worst score gap is 0.6–0.8 nats at Phase-A write points but 0.23–0.27 nats at $F' \geq 10$. A per-step score therefore gives little credit for retention under a lossy recursive policy: keeping a fact at step t is

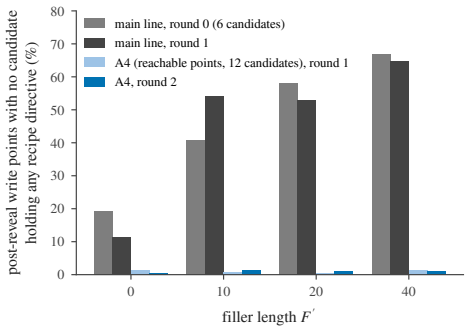

Figure 4: Credit assignment under lag: fraction of post-reveal training write points at which no candidate state holds any recipe directive, per F' (main line, rounds 0–1; A4, rounds 1–2). Once a rewrite has dropped the recipe, best-of-n has nothing to select; A4's reachability filter removes the case by construction. Coverage of the best and worst candidate: Figure 8.

worth something only if it is also kept at $t{+}1, \ldots, t{+}40$, and the current writer does not keep it. More data of the same kind does not change either effect.

The two remedies we tried confirm the diagnosis without removing it. Policy iteration makes retention slightly more visible: in the round-1 candidates the score prefers the more complete candidate 35 % against 26 %. Reachability-restricted sampling (A4) removes the inherited-input case by construction (1 %) and restores a strong preference signal, which is why it is the selected line, but its lag κ stays at the untrained level on validation ($+3$ % [-18, +29] at $F' \in \{20, 40\}$ for the selected seed, -3 % [-19, +20] for the three seeds pooled) and on test. A hand-written retention rule does not face this problem: it is applied at every rewrite, so it keeps a fact for as long as the rule names it, and its reduction of regret does not decay (20–50 % in every rewrite bin on test; Figure 3). What is missing is an objective that credits keeping a fact at t with its value beyond the current writer's forgetting horizon (§7).

6 Limitations

Our environment is a controlled diagnostic: Phase A is scripted, and under protocol F' the writer knows the recipe matters, so F' tests retention of a known-relevant fact rather than the prospective choice of what will matter. GUIDE, which uses task knowledge, remains the strongest budgeted state, and DSSR helps only at short and medium delay. κ is relative to a best-of-eight hindsight compression and a behaviour-cloned reader, and the second reader misses its pre-registered bar by one game. We study only 4B-parameter models, long delays on Phase-A facts are rare, validation selection picked the weakest test seed, and replicates on a second GPU differ by up to 14 points, so each table uses one GPU.

7 Conclusion and Future Work

With a fixed budget, a constant-context agent loses mainly through write-time regret, not the budget, and the regret grows with the number of rewrites a fact must survive. A candidate state is worth what the reader can do from it after the writer has rewritten it: this recursive score predicts return where the fixed-state approximation does not, and a writer trained on it keeps recently observed facts, with a significant gain on held-out test games when facts are needed soon. The gain shrinks with delay, because a per-step objective cannot credit keeping a fact whose payoff lies beyond the current writer's forgetting horizon; a hand-written rule that uses task knowledge still does better there. Future objectives should score a candidate under a continuation that keeps what the candidate keeps, for example by enforcing retention in the roll-forward, training writer and roll-forward jointly, or following a curriculum from short to long delays; readers that can request facts back are the alternative on the reader side. The decomposition, the protocol and the pre-registered evaluation give a way to measure these proposals.

REFERENCES

Abdelrahman Abouelenin et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-LoRAs. *arXiv preprint arXiv:2503.01743*, 2025.

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and William L. Hamilton. Learning dynamic belief graphs to generalize on text-based games. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Leonard Adolphs and Thomas Hofmann. LeDeepChef: Deep reinforcement learning agent for families of text-based games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. TextWorld: A learning environment for text-based games. In *Computer Games Workshop at ICML/IJCAI*, 2018. arXiv:1806.11532.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations (ICLR)*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

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

Haitao Jiang, Lin Ge, Hengrui Cai, and Rui Song. PABU: Progress-aware belief update for efficient LLM agents. *arXiv preprint arXiv:2602.09138*, 2026.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

Minki Kang et al. ACON: Optimizing context compression for long-horizon LLM agents. *arXiv preprint arXiv:2510.00615*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

Junfeng Liao, Qizhou Wang, Jianing Zhu, Bo Du, Rui Yan, and Xiuying Chen. Belief memory: Agent memory under partial observability. *arXiv preprint arXiv:2605.05583*, 2026.

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, 2024.

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

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

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

Joykirat Singh, Zaid Khan, Archiki Prasad, Justin Chih-Yao Chen, Akshay Nambi, Hyunji Lee, Elias Stengel-Eskin, and Mohit Bansal. Agent-BRACE: Decoupling beliefs from actions in long-horizon tasks via verbalized state uncertainty. *arXiv preprint arXiv:2605.11436*, 2026.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon LLM agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025.

Qingyue Wang, Liang Ding, Yanan Cao, Zhiliang Tian, Shi Wang, Dacheng Tao, and Li Guo. Recursively summarizing enables long-term dialogue memory in large language models. *arXiv preprint arXiv:2308.15022*, 2023.

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

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Tianjun Zhang, Fangchen Liu, Justin Wong, Pieter Abbeel, and Joseph E. Gonzalez. The wisdom of hindsight makes language models better instruction followers. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025.