
Where to Replay: Change-Locality-Aware Replay Selection for Refreshing Cartridges under Repeated Corpus Changes

Han Shaun Lee
Stanford University
hanl@stanford.edu

Abstract

Cartridges compress a corpus into a small trained KV cache, but corpora change. Prior work established that warm-started refreshes forget unless old self-study examples are replayed, and that replaying everything can cost as much as a rebuild. We ask the narrower open question: at a *fixed* replay budget, does it matter *which* examples are replayed? We implement a change-locality-aware replay allocator that spends the budget on the observable mention-graph neighbourhood of the changed pages, and compare it with random, lexical-proximity and oracle-graph replay of the same size, update-only, full replay, rebuild, no update, BM25 retrieval and full text, using the official Cartridges code and its tutorial model (Qwen3-4B) on versioned synthetic wikis with exact dependency strata. Under a protocol frozen on a development world, on three held-out worlds with three sequential 5% revisions (6,066 questions per method), locality replay is indistinguishable from random replay: +0.4 points on unchanged facts (95% document-cluster CI $[-0.8, +1.6]$) and +1.1 on dependent multi-hop facts ($[-8.1, +11.1]$); even oracle selection does not help. Diagnostics show why: forgetting is spread over the whole corpus rather than concentrated near the change, and preservation follows how many pages the replay covers. A pre-registered test on two fresh worlds confirms the coverage account: replay stratified to touch every unchanged page beats random replay by +2.1 points on unchanged facts ($[+0.6, +3.6]$), while locality again ties random, also at a 20% update fraction. Replay *amount* dominates: every 128-item refresh is worse overall than not updating (−4.3 points), while warm-started full replay is the only refresh that beats the stale cache (31.9% vs. 28.2%) and outperforms a rebuild at under half its training cost. All cartridges remain far below retrieval over the current text (55.1%). We release the protocol, raw per-question outputs and cost accounting.

1 Introduction

Cartridges [Eyuboglu et al., 2025] replace a long context with a small trained KV cache: synthetic “self-study” conversations about the corpus are distilled into a trainable prefix, after which queries attend only to the prefix. The approach amortizes the cost of a corpus over all later queries—but only while the corpus stands still. Real corpora (wikis, runbooks, codebases) change in small increments, and every increment forces a choice: rebuild the cartridge from scratch, continue training it on the changed material alone, or continue training with some of the old self-study examples *replayed*.

Shepard [2026] measured this trade-off on repeated warm-started updates. Update-only continuation is cheap but, after enough revisions, is statistically indistinguishable from never updating at all; replaying *all* old examples recovers quality but costs about as much as a fresh rebuild; a replay fraction of one half sits in between. What that study leaves open is whether the *identity* of the

replayed examples matters when their *number* is fixed. A corpus change is local: a few pages change, and the facts that depend on them (“who leads the team that owns service S ?” when S changes owner) live on a few more. If forgetting during a refresh is likewise local, then replaying examples from the *neighbourhood* of the change should protect dependent and nearby knowledge far better than replaying a uniformly random sample of the same size—and if forgetting is diffuse, it should not.

This paper. We test that hypothesis with the official Cartridges code and the tutorial model (Qwen3-4B [Qwen Team, 2025]) on a versioned synthetic wiki whose dependency graph is known exactly, so that every question can be stratified into *directly changed*, *dependent* (multi-hop answer changed although the question’s own page did not), *unchanged*, *added*, *removed* and *historic-version* facts, and so that an *oracle* selection rule exists as a diagnostic upper bound. The production candidate never sees the oracle graph: it uses only an observable page-mention graph built from the text, plus a diff of the rendered pages. We compare it at a *matched number of replay examples and training steps* against random replay, lexical (TF-IDF) proximity replay, update-only continuation, full replay, no update, a full rebuild, BM25 retrieval over the current text, and the full current text in context, across three sequential revisions and three update fractions, with document-level bootstrap confidence intervals. Our contributions are (i) a versioned-corpus refresh protocol with explicit stale-answer, preservation, dependent-answer and version-selection metrics and itemized refresh cost, (ii) a cheap, oracle-free locality-aware replay allocator, and (iii) an honest measurement of whether replay *selection*—as opposed to replay *amount*—moves these metrics for Cartridges.

2 Related work

Cartridges and context distillation. Cartridges [Eyuboglu et al., 2025] store a long corpus in a small trainable KV cache (a prefix-tuning parameterization [Li and Liang, 2021]) by *self-study*: synthetic two-agent conversations about sampled corpus chunks are distilled into the cache with a top- k logit objective [Snell et al., 2022]. The official implementation (commit ef34ba97) trains the cache with Adam on packed 2048-token micro-batches and initializes it from the KV states of the corpus text. Cartridges at Scale [Hardalov et al., 2026] trains one cartridge per document and rotates them between GPU and storage; Cartridges++ [Anonymous, 2026a] addresses off-context derailment. Neither studies corpus *revisions*.

Refreshing precomputed memory. The closest prior work is Shepard [2026], which measures what it costs to compose, rebuild and correct precomputed memory with Llama-3.1-8B-Instruct. Its Section 5 chains 13 warm-started cartridge updates on LongHealth records [Adams et al., 2024] and sweeps a *replay fraction* $r \in \{0, 0.5, 1\}$: a share r of the earlier sources’ self-study examples is mixed back into each increment. At generation 13, no-replay updates (0.34 GPU-h per increment) sit statistically at the stale-cache floor (+2.5 pts, $p = 0.51$), while full replay (5.35 GPU-h, more than a fresh retrain at 5.03 GPU-h) reaches 0.525 vs. 0.375 for no replay. The paper does not describe *which* examples are replayed (a fraction, not a selection rule), does not label questions by hop, and names “replay fractions between 0 and 0.5” as the most grounded cost lever. Incremental warm starts, replay as a fraction, rebuilding, and newest-version-wins are therefore established baselines; we use them as such and do not claim them. Our question is narrower: at a *fixed* replay budget, does it matter *which* items are replayed?

Replay selection, knowledge editing and influence. Experience replay is the standard remedy for forgetting in continual learning [Rolnick et al., 2019]; selection heuristics there rank by loss, gradient or coverage of the *old* distribution rather than by proximity to the *change*. Knowledge-editing work shows that edits must propagate to multi-hop consequences [Meng et al., 2022, Zhong et al., 2023], which motivates our dependent-answer stratum. Influence functions [Koh and Liang, 2017] would give a principled selection signal but are too expensive at refresh time; our candidate uses only observable text structure (page mentions) and we report its construction cost. Budgeted KV-cache repair [Anonymous, 2026b] selects raw-KV positions to recompute after a prompt edit; it is training-free and operates on uncompressed prompt caches, not on trained cartridges. We found no prior work on locality- or dependency-aware replay selection for trained KV caches (search recorded in EXPERIMENTS.md).

3 Problem and method

Setting. A corpus is a set of pages $\mathcal{P}_k = \{(p, t_k(p))\}$ at version k . A cartridge C_k is a trainable KV prefix of T tokens per layer. Version 0 yields a self-study pool $\mathcal{I}_0$, where each item i is a (user, assistant) conversation generated by the official SelfStudySynthesizer with page $\pi(i)$ as the teacher’s context and with the teacher’s top-20 logprobs stored for distillation. At revision k the observable change set is the text diff $\Delta_k = \{p : t_k(p) \neq t_{k-1}(p)\}$ (added pages included). Items whose page is in Δ_k are stale and are discarded by *every* method; new items $\mathcal{N}_k$ are synthesized for the pages in $\Delta_k \cap \mathcal{P}_k$. A refresh continues training C_{k-1} on $\mathcal{N}_k \cup \mathcal{R}_k$ for a fixed number of epochs, where $\mathcal{R}_k \subseteq \{i \in \mathcal{I}_{k-1} : \pi(i) \notin \Delta_k\}$ is the replay set with $|\mathcal{R}_k| = R$ fixed. All replay policies therefore see the same new items, the same budget R , the same optimizer schedule and the same random stream; they differ only in which R old items they pick.

Observable dependency index. From the current text alone we build an undirected *mention graph* G_k on pages: $(a, b) \in E$ iff the title of b occurs in the body of a or vice versa. On a wiki this is the link structure; it costs one substring scan per page pair (measured: well under a second for ~ 100 pages) and uses no model forward passes. Every item is attributed to its context page $\pi(i)$ (recorded by the synthesizer), so page-level decisions transfer to items.

Locality-aware replay (candidate). Given Δ_k and G_k , let N_h be the pages within h hops of Δ_k excluding Δ_k itself. The budget is split into an *affected-neighbourhood* share $(1-\gamma)R$, spread round-robin over the pages of N_h nearest-first (so each neighbouring page receives roughly equal coverage rather than the largest page dominating), and an *unchanged-control* share γR drawn uniformly from pages outside N_h . If one side has too few items the remainder flows to the other. Hyper-parameters h and γ are tuned on a development world with disjoint entities and question templates and frozen before any test world is run. A variant that adds a measured *uncertain-coverage* signal (the current cartridge’s distillation loss on probe items) is an ablation; its probe forward passes are counted in the refresh cost.

Baselines at the same budget. *Random replay* samples R items uniformly from valid old items. *TF-IDF proximity* ranks unchanged pages by maximum cosine similarity of their text to the changed pages’ new text and fills the budget page by page. *Oracle-graph replay* (diagnostic only) replaces N_h by the pages that the true fact graph marks as read by any question whose answer changed. *Update-only* sets $R = 0$. *Full replay* replays every valid old item. *Rebuild* trains a fresh cartridge from the version- k text initialization on the full version- k pool with the base schedule. *No update* serves C_0 . *RAG* prepends the top-3 BM25 pages of the *current* corpus to the question with no cartridge; *full text* places the entire current corpus in context (implemented as an untrained text-initialized KV cache, i.e. exact prefill of the corpus).

Metrics. For every question at version k we record *current-answer correctness* (normalized containment of the canonical answer or an alias, and *not* of any previous answer, so hedging with both is wrong), the *obsolete-answer rate* (a previous version’s answer and not the current one), *abstention*, and for historic-version questions whether the model returns the old value rather than the current one. *Preservation* is correctness on the unchanged stratum; *version-selection accuracy* is the fraction of (historic, current) pairs about the same fact that are both right. Costs are itemized per refresh: synthesis tokens and seconds for the new items (shared equally by the refresh methods at a revision), index/selection seconds, training micro-batches, tokens, seconds and peak memory, cache tokens and bytes, and inference prompt/generated tokens and seconds; FLOPs are estimated from token counts and labelled as such.

4 Experimental setup

Upstream code and model. All cartridge training uses the official Cartridges repository at commit ef34ba97 (Apache-2.0; equal to main when cloned) with its `train()` loop unchanged: trainable per-layer K/V prefix with the first token frozen, Adam at learning rate 2×10^{-2} , bf16 autocast, FlexAttention, packed 2048-token micro-batches in the upstream truncate packing mode, top-20 logit distillation with 0.99 probability-mass truncation. The model is the tutorial model, Qwen3-4B (HF revision 1cfa9a72, Apache-2.0), in bf16 with thinking disabled. Self-study uses the official

SelfStudySynthesizer with the tutorial’s five seed-prompt types; because the upstream synthesizer requires a Tokasaurus/SGLang server for top- k logprobs, we implemented an in-process HuggingFace generate backend that returns the same ClientSample objects (text, token ids, top-20 logprobs). This is a backend substitution; the prompts and synthesizer logic are upstream.

Faithful baseline. Before any new experiment we ran the tutorial training on the official synthesized dataset (hazyresearch/arxiv_synthesize_qwen-qwen3-4b_n8192-0) with the official evaluation set, measured peak memory and throughput on the selected GPU, and recorded the distillation loss before and after training (Section 5). This establishes that the training path works as released, not that we reproduce any number from the paper, which used different corpora.

Versioned synthetic wiki. A world is a fictional engineering wiki with pages for services (48), teams (10), people (10), regions (8) and databases (16) plus a change-log page; each page holds one passage per fact (5 service facts, 2 team, 2 region, 2 database, 1 person; 318 mutable facts, ~ 93 pages). Six chains anchored on services define 2- and 3-hop questions (e.g. lead of owning team; city of deployment region; title of the owning team’s lead). Each revision changes a fraction $f \in \{1\%, 5\%, 20\%\}$ of mutable facts: replacements (70%), decommissions of services (15%) and additions of new services (15%) when the change count allows. Passage surface forms are chosen by hash so unchanged pages are byte-identical across versions and the rendered-text diff recovers the changed-page set exactly (tested). The change-log page records every change with its old value, so historic-version questions are answerable from the *current* corpus and a method is never rewarded for erasing provenance. Development and test worlds use disjoint entity-name pools and disjoint question-template sets (tested); the development world (seed 101) is used for all tuning, test worlds (seeds 1–3) only under the frozen protocol.

Questions and strata. Every fact and every chain yields one question per version from the split’s templates; replacements additionally yield historic questions (“what was the owning team of S before revision j ?”). Relative to the previous version a question is *direct* (answer changed and its own page changed), *dependent* (answer changed, own page unchanged), *unchanged*, *added*, *removed* (correct answer: decommissioned) or *historic*. Answers are unique names, codes or numbers (no name is a substring of another; tested), scored by normalized boundary-aware containment. Evaluation questions are generated by templates from the fact graph, independently of self-study; we audit 4-gram overlap between each question and every self-study user turn and report its distribution.

Protocol. Base cartridge: T tokens initialized from the version-0 text, trained on n items per page for E_0 epochs. Refresh at revision k : n new items per changed page, replay budget R items, E_r epochs, warm start from the method’s own previous cartridge (each method has its own chain). Rebuild uses the base schedule on the full version- k pool. Hyper-parameters (T, n, E_0, E_r, R , global batch, h, γ) are set on the development world from the pilot measurements and frozen in configs/proto_frozen.json; the exact values are in Table 2. Uncertainty: 95% bootstrap intervals that resample (world, anchor page) clusters, and paired cluster-bootstrap differences on identical question ids between the candidate and each baseline. Compute: one NVIDIA L40S (48 GB) per job on Modal; all costs below are measured on that GPU unless labelled estimated.

5 Results

5.1 Faithful baseline: the released pipeline runs as published

Running the pinned `train()` on the official tutorial dataset (8,192 self-study conversations about the Cartridges paper itself, 5.38M training tokens) with the tutorial configuration (Qwen3-4B, KV cache initialized from the full 20,001-token L^AT_EX source, i.e. a 20,005-token cache of 2.95 GB in bf16, Adam 2×10^{-2} , global batch 32, one epoch = 68 optimizer steps) took 3,543 s on one A100-80GB (1.5 s per 2048-token micro-batch; peak 36.0 GB) and lowered the top- k distillation loss on the official 32-item evaluation set from 2.959 to 2.358. The same configuration fits an L40S for training (peak 35.3 GB in our pilot) but the generation path’s compiled FlexAttention template exceeds the L40S shared-memory limit at this cache length, so the reproduction ran on A100; all benchmark cartridges below use a 1,024-token cache and run on L40S (0.54 s per micro-batch, peak

Table 1: Development-world calibration (world dev_s101, revision $k=1$, 640 questions; one run, no CIs). Top: base cartridge accuracy on the 606 version-0 questions vs. cumulative epochs. Bottom: refresh accuracy (%) by stratum and the macro-average over the four current-fact strata used to freeze E_r .

Base: 4 ep. 25.4%; 8 ep. 31.2%; 16 ep. 34.8%; text-init only 16.7%; full-text ICL 68.3%; no context 0.0% (240-q subsample)									
Method	E_r	unchanged	direct	dependent	added	historic	ALL	macro-4	train s
No update	–	36.6	0.0	13.0	4.5	41.7	32.2	13.6	0
update-only	2	28.3	26.1	47.8	22.7	8.3	27.3	31.2	17
update-only	4	21.7	47.8	43.5	27.3	0.0	22.5	35.1	32
update-only	8	18.0	52.2	52.2	36.4	16.7	20.5	39.7	64
random	2	31.0	39.1	43.5	22.7	16.7	30.2	34.1	23
random	4	29.2	43.5	39.1	31.8	8.3	28.7	35.9	45
random	8	27.5	56.5	34.8	27.3	16.7	27.7	36.5	90
locality	2	29.7	43.5	43.5	27.3	25.0	29.5	36.0	22
locality	4	26.0	56.5	43.5	22.7	8.3	26.4	37.2	44
locality	8	25.5	60.9	43.5	27.3	8.3	26.2	39.3	88

30.0 GB). These are measurements of the released code, not reproductions of paper numbers (the paper’s corpora differ).

5.2 Development calibration

All tuning used the development world only (Table 1; run 20261022T092701Z_calib). The base cartridge was still improving at 16 cumulative epochs (25.4% $\rightarrow$ 31.2% $\rightarrow$ 34.8% on version-0 questions), far below the full text in context (68.3%) but well above its untrained text initialization (16.7%) and the model without context (0%), so the cartridge carries corpus knowledge it did not have before training. A one-revision refresh exposes the expected trade-off: more refresh epochs learn more of the change (update-only direct accuracy 26.1% $\rightarrow$ 47.8% $\rightarrow$ 52.2% at 2/4/8 epochs) and forget more of what did not change (unchanged 28.3% $\rightarrow$ 21.7% $\rightarrow$ 18.0%, against 36.6% for the stale cartridge); 128 replayed items reduce that forgetting. Before any test world was run we froze $E_0=16$ (largest calibrated value) and $E_r=8$, by a rule stated in the experiment log beforehand: maximize the macro-average of the four current-fact strata averaged over the three calibrated methods (33.8/36.1/38.5% at 2/4/8 epochs). Overall accuracy, which is dominated by the unchanged stratum, would have chosen $E_r=2$; at that setting the refresh barely learns the change and forgetting is too small for replay selection to be testable. The development strata other than “unchanged” contain only 12–23 questions, so the calibration cannot by itself rank the replay policies, and we do not use it to.

5.3 Main comparison on held-out test worlds

Three held-out test worlds (seeds 1–3, 5% update fraction, three sequential revisions) were run once each under the frozen protocol (runs p08_test_s{1, 2, 3}_f5_r3_frozen). Pooled over revisions $k = 1..3$ each method answers 6,066 questions: 5,145 unchanged, 219 direct, 90 dependent, 198 added, 198 removed and 216 historic, spread over 297 (world, anchor page) clusters. Tables 3–6 and Figures 1–2 give the full picture; we summarize the findings that bear on the hypothesis.

Primary endpoint: replay selection does not matter at a fixed budget. Locality-aware replay and random replay of the same 128 items are statistically indistinguishable on every stratum (Table 4): locality minus random is +0.4 points on unchanged facts (95% CI $[-0.8, +1.6]$, $n = 5,145$), +1.1 on dependent facts ($[-8.1, +11.1]$, $n = 90$), +0.9 on direct facts ($[-5.1, +6.5]$) and +0.3 overall ($[-0.8, +1.4]$). The hypothesis that change-local replay protects dependent and unchanged knowledge better than random replay is therefore not supported at the precision these three worlds allow; the unchanged-stratum interval rules out a preservation gain larger than about 1.6 points, while the dependent-stratum interval is too wide to rule out a gain of up to ~ 11 points. Even the *oracle* selector, which replays items from the pages the true fact graph marks as affected, is no better than random on dependent facts (locality – oracle = +2.2, $[-7.5, +12.4]$) and is slightly *worse* on unchanged facts (-1.6 , $[-2.7, -0.5]$ relative to locality). Lexical TF-IDF proximity, which concentrates the budget on the most similar pages, preserves unchanged facts worst among the replay

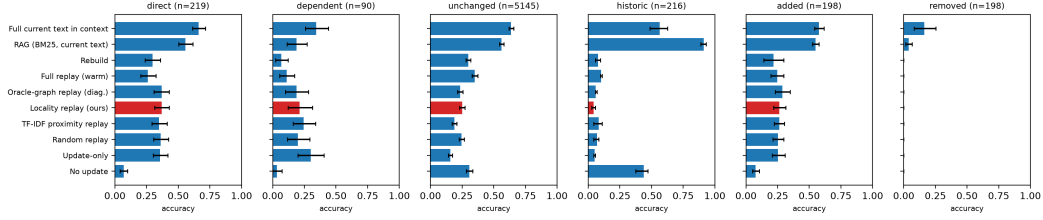


Figure 1: Current-answer accuracy by stratum (revisions $k \geq 1$ pooled over test worlds; bars are 95% document-cluster bootstrap CIs; red = locality replay).

policies (locality – proximity = +6.2, [+4.8, +7.7]). Section 6 explains why: forgetting is not local to the change, so targeting only moves protection around (Figure 3).

Replay amount matters, and every 128-item refresh is worse than not updating overall.

Update-only continuation learns the change (direct 35.6% vs. 6.8% for the stale cartridge) but forgets unchanged facts (15.7% vs. 30.7%). A 128-item replay recovers about a third of that loss (random 24.6%, locality 25.0%), which is not enough: because unchanged facts dominate, every fixed-budget refresh is *below* the stale cartridge on all questions together (locality 23.9% vs. no update 28.2%; paired -4.3 , $[-6.6, -2.1]$). Only full replay of all $\sim 1,300$ valid old items both learns the change and preserves the rest (unchanged 35.0%, overall 31.9%; locality – full replay = -9.9 on unchanged, $[-11.4, -8.5]$). Full replay also beats a rebuild from scratch with the base schedule (overall 31.9% vs. 27.4%) at less than half its training cost (Table 6: 351 s vs. 701 s of training per revision, plus 115 s of update synthesis shared by all refresh methods). Unlike the 13-generation LongHealth chains of Shepard [2026], where full replay cost more than a fresh retrain, here a warm-started full replay is the cheapest configuration that improves on the stale cartridge; the difference is that our rebuild retrains from the text initialization for the full base schedule while full replay continues an already-trained cache for $E_r=8$ epochs.

Over revisions. Unchanged-fact accuracy decays monotonically for every fixed-budget policy (locality 25.5/25.7/23.9%, random 27.2/24.5/22.4%, update-only 17.4/17.2/12.7% at $k = 1, 2, 3$) and is flat for full replay (34.1/35.7/35.1%) (Figure 2). Dependent-fact accuracy is noisy at 30–37 questions per revision; update-only has the highest point estimate (30.0% pooled), consistent with dependent answers being learned from the new self-study of the changed pages rather than protected by replay.

Text baselines dominate. Every cartridge is far below the current text: BM25 retrieval of three current pages reaches 55.1% overall and the full current corpus in context 61.4% (vs. 31.9% for the best cartridge), at the price of $6.6\times$ more prompt tokens per question for RAG and a $5.6\times$ longer prefix for full text. Historic (“what was X before revision j ”) questions are answered by RAG (91.2%, the change-log page is retrieved) and by the stale cartridge (44.0%, it still holds the old value), but by no refreshed cartridge ($\leq 10.2\%$); refreshed cartridges overwrite rather than version their knowledge, and version-selection accuracy (both the historic and the current question about a fact right) is at most 2.8% for any cartridge vs. 80.1% for RAG. No cartridge ever answers that a removed service was decommissioned (0.0%); full text manages 16.2%.

Leakage audit. 6.8% of all evaluation questions share at least half of their word 4-grams with some self-study user turn (unchanged 7.3%, direct 7.1%, dependent 1.9%); since this overlap is the same for every cartridge method trained on the same pool, it cannot explain differences between replay policies.

Evaluation path. At $k = 2$ the final, two-question generation batch of every evaluation fell back to eager FlexAttention after an Inductor shared-memory failure on the L40S (recorded per evaluation in `costs.jsonl`); the fallback hit every method identically, so paired comparisons are unaffected.

Table 2: Frozen protocol (configs/proto_frozen.json).

Parameter	Value
model	Qwen/Qwen3-4b
cache_tokens	1024
n_per_page	16
base_epochs	16
refresh_epochs	8
replay_budget	128
global_batch_size	8
lr	0.02
locality_hops	1
control_frac	0.25
synth_max_tokens_b	384
max_new_tokens	24
rag_k	3
seed	0

Table 3: Current-answer accuracy (%) by stratum after revisions $k \geq 1$, pooled over test worlds; 95% CIs resample (world, page) clusters. [†]oracle uses the true fact graph: diagnostic upper bound only.

Method	direct <i>n</i> =219, 90 docs	dependent <i>n</i> =90, 55 docs	unchanged <i>n</i> =5145, 288 docs	historic <i>n</i> =216, 3 docs	added <i>n</i> =198, 18 docs	removed <i>n</i> =198, 18 docs	ALL <i>n</i> =6066, 297 docs
No update (C_0)	6.8 [4,10]	3.3 [0,7]	30.7 [28,33]	44.0 [38,47]	7.6 [5,11]	0.0 [0,0]	28.2 [26,31]
Update-only ($R=0$)	35.6 [30,42]	30.0 [20,41]	15.7 [14,17]	5.1 [4,6]	25.3 [21,31]	0.0 [0,0]	16.1 [14,18]
Random replay	36.1 [31,42]	20.0 [12,29]	24.6 [23,27]	6.9 [4,8]	25.3 [21,30]	0.0 [0,0]	23.6 [22,26]
TF-IDF proximity replay	34.7 [29,41]	24.4 [16,34]	18.9 [17,21]	8.3 [4,11]	26.3 [22,30]	0.0 [0,0]	18.8 [17,20]
Locality replay (ours)	37.0 [31,43]	21.1 [12,31]	25.0 [23,27]	4.2 [3,6]	26.3 [22,31]	0.0 [0,0]	23.9 [22,26]
Oracle-graph replay [†]	37.0 [31,43]	18.9 [10,28]	23.4 [21,26]	6.0 [6,7]	28.8 [23,35]	0.0 [0,0]	22.6 [21,25]
Full replay (warm start)	26.0 [20,32]	11.1 [5,17]	35.0 [33,37]	10.2 [10,11]	24.7 [20,30]	0.0 [0,0]	31.9 [30,34]
Rebuild from scratch	29.7 [24,36]	6.7 [2,12]	29.8 [28,32]	7.9 [6,10]	21.7 [14,30]	0.0 [0,0]	27.4 [26,30]
RAG (BM25 top-3, current text)	55.7 [50,62]	18.9 [11,27]	56.2 [55,58]	91.2 [89,93]	55.1 [53,58]	4.0 [2,7]	55.1 [53,58]
Full current text in context	66.2 [61,72]	34.4 [26,44]	63.7 [62,66]	56.5 [49,62]	57.6 [54,62]	16.2 [8,25]	61.4 [59,63]

6 Ablations and diagnostics

Selection signal. The oracle-graph and TF-IDF arms are ablations of the selection signal at a fixed budget: replacing the observable mention graph by the true fact graph does not improve dependent-fact accuracy, and a lexical signal that concentrates the budget hurts preservation (Section 5.3).

Replay amount. Update-only ($R = 0$), 128-item replay and full replay ($R \approx 1,300$) span the amount axis at the same epochs.

Number of revisions. Figure 2 shows the per-revision trajectory; the gap between fixed-budget replay and full replay widens with k on unchanged facts.

Is forgetting local? (diagnostic). The hypothesis presupposes that a refresh damages knowledge near the change. We tested this premise directly (scripts/forgetting_locality.py): for each unchanged question we compute the distance, in the same observable mention graph the candidate uses, from its page to the changed pages, and compare every method with the stale cartridge on the same question (Figure 3). Forgetting is *diffuse*: update-only loses 16.3 points (95% CI [13.3, 19.4]) at distance 1, 18.7 [16.0, 21.5] at distance 2 and 23.1 [6.1, 40.6] at distance ≥ 3 , and nothing on the changed pages themselves (-2.5 , $[-6.6, +1.4]$), whose unchanged facts are re-taught by the new self-study. Because the wiki’s link graph is dense, 86% of unchanged questions lie at distance 1–2. Targeted replay therefore *redistributes* protection: locality – random is $+2.1$ $[+0.1, +4.2]$ at distance 1 and -1.5 $[-3.0, +0.1]$ at distance 2; oracle – random is $+2.0$ $[-0.2, +4.2]$ and -4.9 $[-6.8, -3.1]$; TF-IDF proximity, which spends the budget on about eight pages, is worse than random at every distance (-8.2 at distance 1). Full replay protects all distances ($+10.7$ to $+12.3$ over random at distances 1–2). Targeting a neighbourhood cannot help when the damage is spread over the whole corpus.

Table 4: Paired difference in accuracy (percentage points, locality minus baseline) on identical question ids, with 95% document-cluster bootstrap CIs. Positive favours locality replay.

Locality –	direct	dependent	unchanged	historic	added	removed	ALL
No update (C_0)	+30.1* [+24,+37]	+17.8* [+8,+29]	-5.7* [-8,-4]	-39.8* [-43,-35]	+18.7* [+13,+24]	+0.0 [+0,+0]	-4.3* [-7,-2]
Update-only ($R=0$)	+1.4 [-4,+7]	-8.9 [-20,+2]	+9.3* [+8,+11]	-0.9 [-3,+1]	+1.0 [-4,+6]	+0.0 [+0,+0]	+7.8* [+7,+9]
Random replay	+0.9 [-5,+6]	+1.1 [-8,+11]	+0.4 [-1,+2]	-2.8 [-6,+0]	+1.0 [-5,+7]	+0.0 [+0,+0]	+0.3 [-1,+1]
TF-IDF proximity replay	+2.3 [-4,+9]	-3.3 [-13,+7]	+6.2* [+5,+8]	-4.2 [-8,+1]	+0.0 [-4,+5]	+0.0 [+0,+0]	+5.1* [+4,+6]
Oracle-graph replay [†]	+0.0 [-6,+6]	+2.2 [-8,+12]	+1.6* [+1,+3]	-1.9 [-3,+0]	-2.5 [-7,+3]	+0.0 [+0,+0]	+1.3* [+0,+2]
Full replay (warm start)	+11.0* [+5,+17]	+10.0 [+0,+21]	-9.9* [-11,-8]	-6.0* [-7,-4]	+1.5 [-6,+9]	+0.0 [+0,+0]	-8.0* [-9,-7]
Rebuild from scratch	+7.3 [-0,+14]	+14.4* [+3,+27]	-4.7* [-7,-3]	-3.7 [-6,+0]	+4.5 [-4,+13]	+0.0 [+0,+0]	-3.5* [-5,-2]
RAG (BM25 top-3, current text)	-18.7* [-26,-12]	+2.2 [-11,+16]	-31.2* [-33,-29]	-87.0* [-90,-83]	-28.8* [-34,-24]	-4.0* [-7,-2]	-31.3* [-34,-28]
Full current text in context	-29.2* [-35,-23]	-13.3 [-25,+0]	-38.7* [-41,-37]	-52.3* [-58,-46]	-31.3* [-35,-28]	-16.2* [-25,-8]	-37.5* [-39,-35]

Table 5: Obsolete-answer rate (%): the model returns a previous version’s value (for historic questions: the current value instead of the asked-for old one).

Method	direct <i>n</i> =219, 90 docs	dependent <i>n</i> =90, 55 docs	unchanged <i>n</i> =5145, 288 docs	historic <i>n</i> =216, 3 docs	added <i>n</i> =198, 18 docs	removed <i>n</i> =198, 18 docs	ALL <i>n</i> =6066, 297 docs
No update (C_0)	27.9 [22,34]	16.7 [9,25]	2.0 [1,3]	1.4 [0,4]	0.0 [0,0]	20.2 [12,29]	3.7 [3,5]
Update-only ($R=0$)	4.1 [1,8]	2.2 [0,6]	0.9 [0,1]	35.6 [32,40]	0.0 [0,0]	11.1 [6,17]	2.6 [1,4]
Random replay	5.5 [3,8]	6.7 [2,13]	1.3 [1,2]	43.5 [40,47]	0.0 [0,0]	15.7 [12,19]	3.4 [2,5]
TF-IDF proximity replay	4.1 [2,7]	7.8 [3,13]	0.8 [0,1]	40.7 [32,49]	0.0 [0,0]	13.1 [9,18]	2.8 [1,5]
Locality replay (ours)	8.2 [4,13]	8.9 [3,15]	1.3 [1,2]	41.2 [38,43]	0.0 [0,0]	21.2 [16,27]	3.7 [2,5]
Oracle-graph replay [†]	7.8 [5,12]	12.2 [6,19]	1.1 [1,2]	42.1 [40,44]	0.0 [0,0]	15.2 [9,22]	3.4 [2,5]
Full replay (warm start)	9.6 [6,14]	17.8 [10,26]	1.3 [1,2]	44.9 [36,51]	0.0 [0,0]	22.7 [18,28]	4.1 [2,6]
Rebuild from scratch	5.0 [2,8]	8.9 [4,15]	1.2 [1,2]	43.5 [39,50]	0.0 [0,0]	14.6 [11,19]	3.4 [2,5]
RAG (BM25 top-3, current text)	5.0 [3,8]	1.1 [0,4]	0.4 [0,1]	1.4 [0,3]	0.0 [0,0]	18.2 [14,23]	1.2 [1,2]
Full current text in context	5.0 [2,9]	3.3 [0,8]	1.8 [1,3]	28.2 [25,31]	0.0 [0,0]	32.8 [26,39]	3.9 [2,6]

Protection follows replay coverage (diagnostic). To see *which* pages each policy protected we recovered the source page of every replayed item from the exact training sets of all 36 fixed-budget refreshes (`scripts/replay_coverage.py`). With 128 items over ~ 81 unchanged pages per revision, locality replay covers 74.2 pages on average, random 64.4, oracle 51.1 and TF-IDF proximity 8.0. Splitting unchanged questions by whether their page was covered by a policy and/or by random replay gives a clean picture: where only the policy covered the page it beats random (locality +6.1 [+2.8, +9.9], $n = 775$; oracle +13.6 [+8.1, +19.5]; proximity +15.0 [+5.9, +25.5]), where only random covered it the policy loses (locality -6.0 [-11.1, -1.5], $n = 451$; oracle -5.9 [-7.8, -4.1]; proximity -9.4 [-11.1, -7.9]), and where both or neither covered it the difference is near zero (locality -0.3 and +1.2). A policy’s net effect on preservation is thus its coverage surplus over random: locality’s +0.4 is 775 vs. 451 questions on pages covered by only one of the two; oracle and proximity are worse because they concentrate the budget on fewer pages. This yields a falsifiable prediction: replay stratified to cover every unchanged page should beat random (Section 6).

A coverage-based alternative, pre-registered on fresh worlds. Because the coverage account was derived on the test worlds, we tested it out of sample. Before launch we committed a protocol (`configs/proto_prereg_stratified.json`) adding *stratified replay*: the same 128 items spread round-robin over all valid old pages in random order, so every unchanged page receives at least one item; it uses no change information. It ran under the frozen hyper-parameters on two *fresh* test worlds (seeds 4 and 5, 5%, three revisions) generated after all other results; the registered primary endpoint was stratified – random on unchanged facts with success iff the 95% cluster-bootstrap lower bound exceeds zero (a third world was dropped before launch for budget). The endpoint was met: +2.05 points [+0.60, +3.58] ($n = 3,455$, 192 documents; all 81 unchanged pages covered per revision vs. 66.5 for random and 73.5 for locality), with +2.2 [+1.0, +3.6] overall, +5.6 [0.0, +11.9] on direct and +16.2 [+5.6, +28.2] on dependent facts ($n = 37$). The v1 null replicated on the same fresh worlds: locality – random = +0.03 [-1.33, +1.38] on unchanged facts. The gain is real but small relative to the amount effect: stratified replay is still below the stale cartridge overall (24.1% vs. 27.4%, paired -3.2 [-5.6, -1.0]), so at this budget no selection rule substitutes for replaying more.

Larger changes (20% update fraction). Two pre-generated 20% test worlds (seeds 1 and 2; ~ 50 changed pages per revision) were run under the frozen hyper-parameters with update-only, random, locality and oracle replay plus no update, RAG and full text (rebuild and full replay omitted for

Table 6: Measured refresh cost per revision (mean over worlds and revisions, L40S): items trained, 2048-token micro-batches, training seconds, synthesis seconds attributed (update items shared equally by the refresh methods), selection seconds, peak training memory, cache tokens, and mean inference prompt tokens per question batch. Estimated PFLOP = $6NT$ over trained tokens plus $2N$ per synthesized token ($N = 4.02B$), labelled estimate.

Method	new	replay	micro-b.	train s	synth s	select s	peak GiB	cache tok	est. PFLOP
No update (C_0)	—	—	—	—	—	—	—	1024	—
Update-only ($R=0$)	258	0	156	65	19	0.00	28.1	1024	8.0
Random replay	258	128	235	92	19	0.00	28.1	1024	12.2
TF-IDF proximity replay	258	128	233	91	19	0.01	28.1	1024	12.0
Locality replay (ours)	258	128	228	90	19	0.00	28.1	1024	11.8
Oracle-graph replay [†]	258	128	229	91	19	0.00	28.1	1024	11.9
Full replay (warm start)	258	1294	933	351	19	0.00	28.2	1024	48.7
Rebuild from scratch	—	—	1865	701	115	—	28.1	1024	98.0
RAG (BM25 top-3, current text)	—	—	—	—	—	—	—	0	—
Full current text in context	—	—	—	—	—	—	—	5720	—

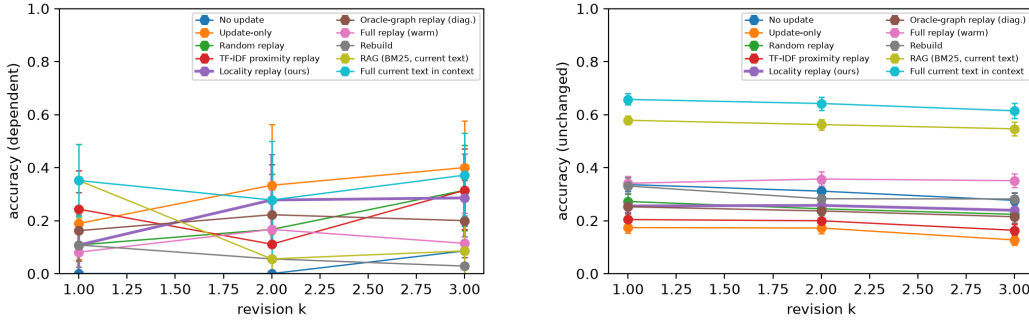


Figure 2: Accuracy over sequential revisions for the dependent (left) and unchanged (right) strata.

budget; the base cartridge is reused from the identical version-0 corpus of the 5% runs). With 5,484 questions per method the dependent stratum grows to 133. Locality – random is $+1.3$ $[0.0, +2.6]$ on unchanged, -3.0 $[-9.2, +3.1]$ on dependent and $+0.6$ $[-0.3, +1.5]$ overall; oracle selection is indistinguishable from locality. At this change rate the one-hop neighbourhood contains essentially every unchanged page, so locality’s round-robin covers all 57.8 unchanged pages per revision (random: 53.5) and behaves like stratified replay, consistent with the coverage account. All fixed-budget refreshes end at the stale cartridge’s level overall (locality 17.6% vs. 16.6%, paired $+1.0$ $[-3.7, +5.2]$): they learn the change (direct 25.9% vs. 6.7%) and forget as much as they gain; RAG (45.0%) and full text (52.1%) remain far ahead.

Robustness to leakage and scoring. `scripts/robustness.py` recomputes the paired comparisons (i) after removing every question with $\geq 50\%$ 4-gram overlap to any self-study user turn and (ii) with strict exact match instead of containment. Strict and containment scoring agree on 98% of rows, and no method ever hedges by naming both the current and a previous answer on changed facts. On unchanged facts locality – random is $+0.4/ +0.6/ +0.5$ points (containment/filtered/strict; all CIs include zero), full replay – random $+10.3/ +9.9/ +10.0$, and the pre-registered stratified – random $+2.1/ +2.1/ +2.0$ with lower bounds $+0.6/ +0.6/ +0.7$. No conclusion depends on the scoring rule or on the overlapping questions.

7 Limitations

Synthetic corpus. The wiki is synthetic by design (exact dependency graph, unambiguous answers, exact strata); natural documents have noisier link structure, paraphrastic facts and ambiguous answers, and we make no claim about transfer beyond what the observable-signal design permits. **One model, one cache size.** All cartridge results use Qwen3-4B with a 1024-token cache; the tutorial reproduction uses the tutorial’s full-text initialization. **Short self-study items and a small corpus.**

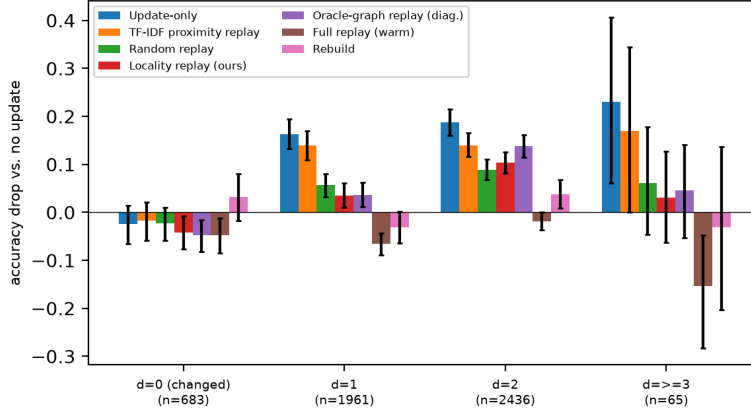


Figure 3: Accuracy drop on unchanged facts relative to the stale cartridge, by mention-graph distance between the question’s page and the changed pages (test worlds, $k = 1..3$ pooled; 95% cluster-bootstrap CIs). Forgetting under update-only is flat in distance; targeted replay shifts protection toward distance 1 at the expense of distance 2.

Our pages are short, so items pack densely and training is cheap; cost ratios between methods are the meaningful quantity, not absolute seconds. **Synthesizer backend.** We used the official synthesizer prompts with an HF generate backend instead of Tokasaurus; distributions should match in expectation but kernel-level numerics differ. **Hyper-parameters.** The refresh schedule was calibrated on one development world; test worlds use the frozen protocol, but a different budget R or epoch count could change the ranking, which we probe in ablations only within budget. **Statistical power.** The dependent stratum is small by construction (only hub facts create many dependents); intervals are correspondingly wide and we report them rather than hide them. **Seeds and power.** Each test world was trained once (seed 0); world-to-world variation is part of the cluster bootstrap but optimizer-seed variation within a world is not separately estimated. Pooled over three worlds the dependent stratum has only 90 questions, so the primary dependent-fact comparison cannot exclude effects of up to ~ 11 points. **Update fraction.** The main comparison uses 5%; 20% was run on two worlds with a reduced set of arms (no rebuild or full replay) and 1% was not run (its worlds have almost no dependent questions). The pre-registered stratified test used two fresh worlds; a third was dropped before launch for budget. **Rebuild cost.** Our rebuild reuses the cached self-study items of unchanged pages (synthesis is keyed by page content), so its cost counts only the update synthesis; a rebuild that re-synthesizes everything would add ~ 11 – 13 minutes of generation per revision. **Evaluation kernel.** One small batch per evaluation at $k = 2$ ran with eager instead of compiled FlexAttention (identically for all methods). **Compute.** All numbers come from single-GPU runs within a 40 USD budget (provider-billed spend in `cost_ledger.csv`).

8 Reproducibility statement

Code, configurations, raw per-question outputs, cost logs and the paper source are in the project repository. Upstream code is pinned to HazyResearch/cartridges@ef34ba97 and used unchanged for cache initialization, self-study prompting, dataset packing, training and generation; our additions are the HF generation backend for the synthesizer, a file-based warm-start initializer, the corpus generator, the replay policies and the evaluation harness. Model and dataset revisions, licenses and hashes are in `sources.lock.json`; Python dependencies in `requirements.lock.txt`; every GPU job records its config, git commit, dirty-diff hash, hardware, software versions, wall-clock, estimated cost and exit status in `results/p08/<run_id>/`, and every number in this paper is produced by `scripts/aggregate.py`, `scripts/make_tables.py` and `scripts/make_figures.py` from those raw files. Synthetic worlds are regenerated deterministically by `scripts/make_worlds.py` (seed, split, update fraction, revisions); their SHA-256 is stored in each run record. The protocol was frozen (`configs/proto_frozen.json`) after development-world calibration and before the first test-world run; development and test worlds share

no entity names or question templates (tested in `tests/`). All compute ran on single NVIDIA L40S GPUs on Modal; the total cost is listed in `cost_ledger.csv` and in the appendix.

References

- Lisa Adams, Felix Busch, Tianyu Han, et al. Longhealth: A question answering benchmark with long clinical documents. *arXiv preprint arXiv:2401.14490*, 2024.
- Anonymous. Cartridges++: Kv cache compression without off-context derailment. *arXiv preprint arXiv:2609.35621*, 2026a.
- Anonymous. Budgeted kv-cache repair for evolving contexts. *arXiv preprint arXiv:2610.17338*, 2026b.
- Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*, 2025.
- Momchil Hardalov, Gonzalo Iglesias, and Adrià de Gispert. Cartridges at scale: Training modular kv caches over large document collections. *arXiv preprint arXiv:2606.04557*, 2026.
- Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *ICML*, 2017.
- Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *NeurIPS*, 2022.
- Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *NeurIPS*, 2019.
- Asa Shepard. What it costs to compose, rebuild, and correct precomputed memory. *arXiv preprint arXiv:2608.30647*, 2026.
- Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.
- Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions. *EMNLP*, 2023.

A Additional details

A.1 Cost-accounting boundary

A refresh at revision k is charged for: (i) synthesis of self-study items for the pages in the observable diff (prompt and generated tokens, wall-clock on the same GPU with the HF backend; shared equally by the refresh methods, since all of them consume the same new items), (ii) index construction and replay selection (wall-clock; the mention graph needs no model calls), (iii) cache optimization (number of 2048-token micro-batches, tokens per epoch, wall-clock, peak allocated memory), and (iv) the resulting cache size (tokens, bf16 bytes). Inference cost is reported separately as prompt and generated tokens and wall-clock per evaluation pass. Rebuild is charged (i) for the same new items plus (iii) with the base schedule over the whole version- k pool. RAG is charged for its BM25 index build and for its longer prompts; full-text-in-context is charged for exact prefill of the corpus (reported as cache bytes of the untrained text cache). FLOPs are *estimated* as $6N_{\text{params}} \cdot \text{tokens}$ for training tokens and $2N_{\text{params}} \cdot \text{tokens}$ for synthesis/inference tokens; they are labelled as estimates wherever they appear. Model loading and FlexAttention compilation are amortized across all runs in a container and excluded from per-refresh costs; the pilot reports them separately.

A.2 Upstream code observation

`TrainableCache.from_pretrained` in the pinned commit infers the number of frozen prefix tokens from `frozen_keys[0].size(1)` (the head axis) rather than `.size(2)` (the token axis). With the default of one frozen token and eight KV heads, a cache reloaded through that path would freeze eight tokens. The upstream training entry point initializes from text and never hits this path, so the faithful baseline is unaffected; our warm starts use a loader that reads the token axis.

A.3 Paraphrase-overlap audit

For every evaluation question we compute the maximum fraction of its word 4-grams that occur in any self-study *user* turn of the pool the cartridge was trained on (version-matched). Evaluation questions are template-rendered from the fact graph and never shown to the synthesizer; overlap therefore measures how often the teacher model spontaneously asked a near-identical question. The distribution is reported in Section 5.

A.4 Synthesizer backend

The upstream `SelfStudySynthesizer` issues two batched chat calls per context chunk (bot A generates a user turn from a seed prompt; bot B answers with the chunk in its system prompt and returns top-20 logprobs). Our backend implements the upstream `Client` interface with `HuggingFace` `generate`; concurrent per-page requests with identical generation parameters are coalesced into one GPU batch, and a `LogitsProcessor` records the top-20 log-probabilities of the raw next-token distribution at every step (undoing the temperature warper for bot A). Each conversation is tagged with the id of the page it was generated from, which is the only item-to-page attribution the replay policies use.