Asia Pacific Science Press

Asia Pacific Economic and Management Review
ISSN(O):3005-9275
ISSN(P):3005-9267
Vol. 3 No.4 (2026)

Compression Remembers the Detour: How Context Compression Amplifies Off-Task Content and Derails LLM Agents

Zhan Zhang[1,2]* , Wenzhi Zhang[2]

1. Ernst & Young, Luohu District, Shenzhen, Guangdong, 518000, China

2. Nanjing University of Finance and Economics, Nanjing, Jiangsu, 210023, China

Corresponding author: Zhan Zhang

Abstract: Long-horizon LLM agents rely on context compression to keep growing histories within a finite window [1][2]. These mechanisms are goal-agnostic—retention follows volume, recurrence, and generic salience rather than task relevance—and we show they systematically amplify off-task content, inducing Compression-Induced Goal Drift (CIGD). We formalize CIGD via the drift amplification ratio (DAR), evaluate six compression strategies on a controlled injection benchmark (n=150 per cell), and propose Goal-Anchored Compression (GAC): a pinned goal anchor, a negation ledger, status-aware retention scoring, and drift re-anchoring. Volume- and recurrence-based compressors amplify detours monotonically with repetition (attention-scored eviction reaches DAR=1.33 at twelve-fold repetition), while query-conditioned compressors de-amplify. Under severe budgets, summarization and scored eviction lose the user's negated constraint in 98% and 97% of episodes where GAC retains it in 100%; GAC eliminates residual detour content (DAR=0) with 3.6× smaller contexts. Prior work asks what compression forgets [3][4]; we ask what it amplifies.

Keywords: Context Compression; Large Language Model Agents; Goal Drift; Agent Memory; Goal-Anchored Compression

Published: Sep 5, 2026

DOI: https://doi.org/10.62177/apemr.v3i4.1704

1. Introduction

1.1 Background and Motivation

Long-horizon LLM agents must sustain coherent behavior over hundreds of steps [14][15][42][43][45][48], yet their working context remains finite. The mismatch is resolved by context management [16][17][18][19][20][21][22][23][24][29][30][31][32][33][37]—truncation, summarization, selective eviction, or off-loading to external memory [1][2][5][34][35][36][47][49][50]—all sharing one unexamined assumption: what survives compression is what the agent will treat as its history and, implicitly, as its instruction. Yet the survival scores are goal-agnostic: recency, relevance, and generic "importance" [6], accumulated attention [7], and salience-by-similarity [8] measure how much text an event generated, not whether it should govern future behavior.

The motivating vignette

An anonymized case from our case-study corpus (§7): a user asks a cooking assistant for a tomato-and-egg stir-fry. Mid-planning, the agent hypothesizes adding pork—an unprompted digression—and spends several turns on cuts, marinades, and cooking times. The user interrupts: "No pork. Just tomato and egg." The agent apologizes and resumes. From the user's

perspective the trajectory is repaired; from the context manager's it is not: the pork digression generated the most tokens (volume), was restated across turns (recurrence), and accumulated outsized attention mass. When the window overflows, the summarizer keeps a prominent pork plan while the short rejection is dropped; the agent, conditioning on a memory in which pork is prominent and unrejected, re-introduces pork two steps later. Compression remembered the detour and forgot the verdict.

1.2 Research Gap and Problem Statement

Figure 1. The CIGD phenomenon on the cooking vignette. The interaction pattern: the agent dwells on a pork addition the user has explicitly rejected, producing bulky, recurrent off-task text; goal-agnostic compression keeps the detour while dropping the rejection, and the post-compression policy re-invokes the detour.

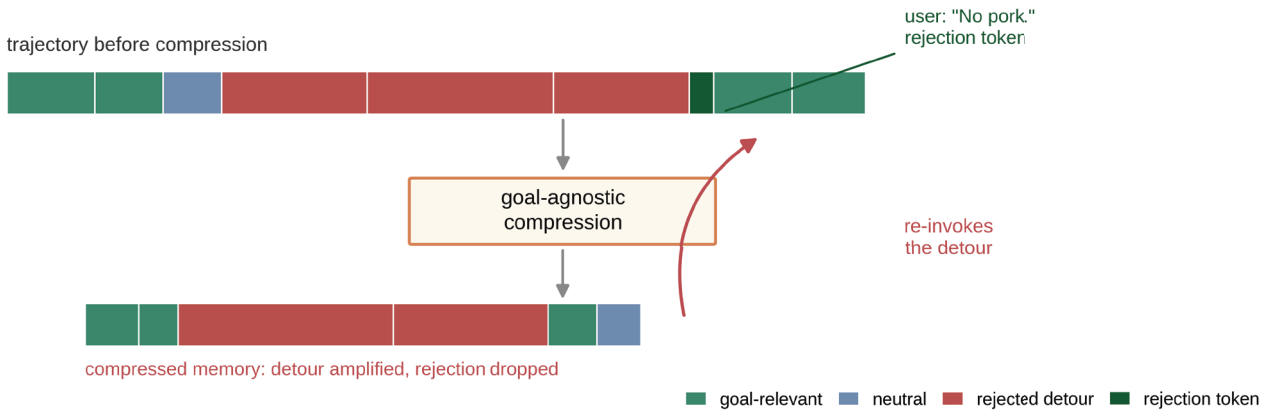
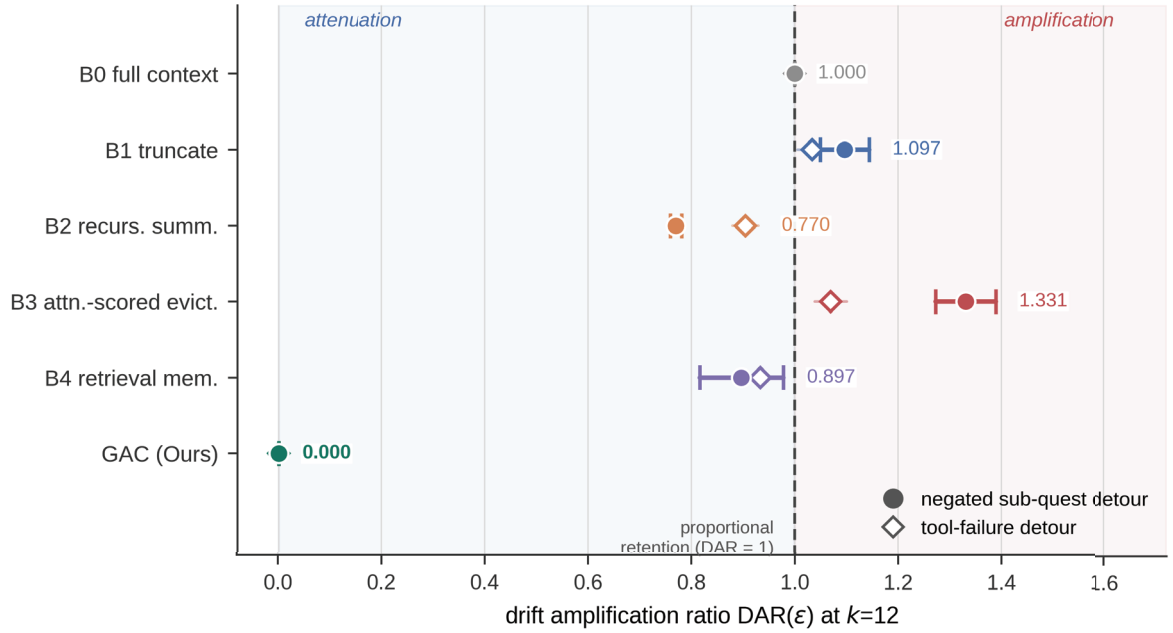


Figure 2. Measured drift amplification ratio at twelve-fold detour repetition (n=150 per cell): recurrence-scored eviction (B3) and truncation (B1) amplify the rejected detour above its original share on both detour types (solid dots: negated sub-quest; hollow diamonds: tool-failure); GAC retains none of it.



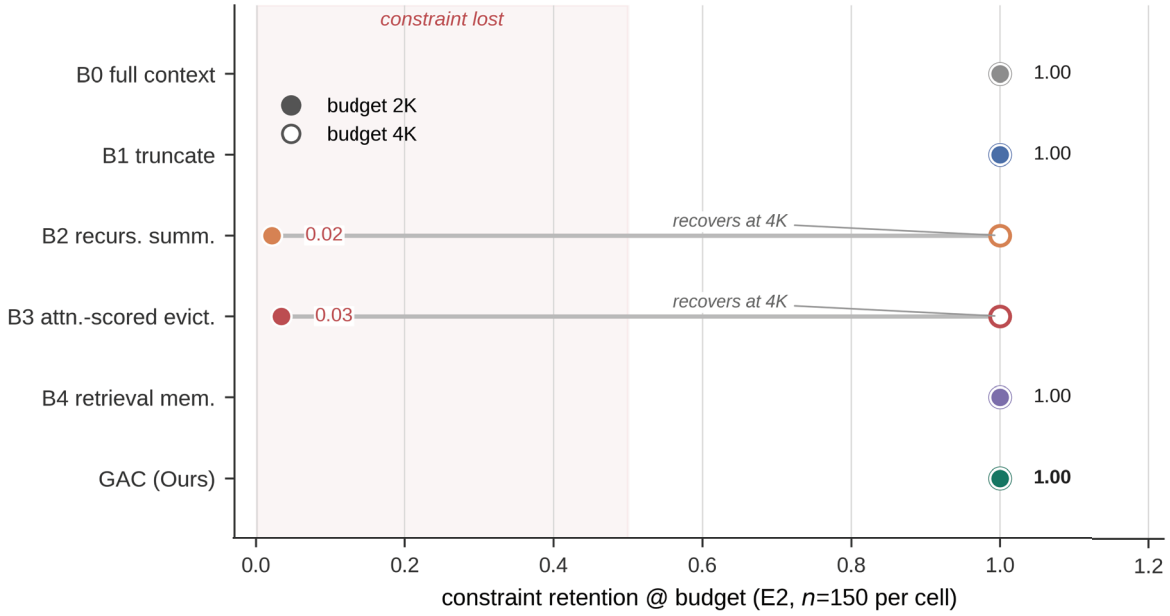
Prior work asks what compression forgets; we ask what it amplifies. Governance Decay showed across 1,323 runs that compaction silently erases governance constraints, violations rising from 0% to 30% (up to 59%) [3]; Lost in Compaction found only 17% average retention of injected constraints [4][9]. Both treat compression as subtractive. The complementary, additive failure—off-task content over-represented by compression, steering behavior—has not been isolated: forgetting studies quantify what leaves memory, not what dominates it, and goal-drift measurements [10] do not model the compression

step that manufactures the drifting patterns.

We call this phenomenon Compression-Induced Goal Drift (CIGD). Off-task episodes are disproportionately bulky and recurrent—a failed tool-call loop generates a call, an error, a retry, and a rationalization per iteration—while on-task progress is often terse. When retention weight is driven by volume and recurrence [7][6], compression systematically raises the off-task share of effective memory. CIGD is present when DAR exceeds 1 and the amplified residue steers subsequent actions.

A second mechanism compounds the first: compression strips discourse status. Summaries preserve propositional content but drop pragmatic labels—rejected, superseded, resolved, active—recording that pork was discussed, not that pork was rejected. We call the record of user rejections the negation ledger; §3 shows it is among the first casualties of goal-agnostic compression.

Figure 3. Constraint retention at the two tightest budgets: recursive summarization (B2) and scored eviction (B3) lose the user's negated constraint in 97–98% of episodes at 2K (solid dots) and recover it only when the budget relaxes to 4K (hollow dots); GAC never loses it. Full quantitative results appear in §3 (Figures 5 and 6) and §5 (Figure 8).



2. Compression-Induced Goal Drift: Definition and Measurement

2.1 Problem Formulation

2.1.1 Agent trajectories, compression operators, and retention weights

We model a long-horizon episode as a trajectory $\tau=(e_1,\ldots,e_n)$, where each event e_i (user instruction, reasoning step, tool invocation, observation) carries token mass $tokens(e_i)$, discourse status $s(e_i)$ (§3.1.3), and a reference count $rec(e_i)$ of later re-invocations, conditioned on a user goal g. When τ exceeds the working-context budget B, a compression operator C maps it to $C(\tau)$ with $\text{tokens}(C(\tau))\leq B$. All widely deployed operators select or rewrite events by an implicit retention weight $w(e)$, well approximated by

$$w(e) = \lambda \cdot \text{vol}(e) \cdot \text{rec}(e) \cdot \text{sal}(e), \tag{1}$$

where $\text{vol}(e) = \text{tokens}(e)/\sum_{e'\in\tau} \text{tokens}(e')$ is relative token mass, $\text{sal}(e)$ a model-internal salience score, and λ a budget-dependent normalizer. Truncation is the degenerate case with positional $w(e)$; summarization instantiates $sal(e)$ through a learned prior over what "reads important." Crucially, none conditions on the goal. We call C goal-agnostic when

$$\frac{\partial w(e)}{\partial \,\text{rel}(e,g)} = 0 \quad \text{for all} e \in \tau, \tag{2}$$

2.1.2 Two salience biases and the drift amplification ratio

Equation (1) exposes two structural biases: salience-by-volume (more tokens, higher weight) and salience-by-recurrence (restated events accumulate weight). Both have independent support: attention weight is a noisy importance proxy [13]

concentrating on semantically empty tokens [25], and repetition is self-reinforcing [11][12]. Detours are the worst-case input: tool-failure loops generate volume by construction and force re-reading on every retry; rejected sub-quests generate both through negotiation and repeated returns.

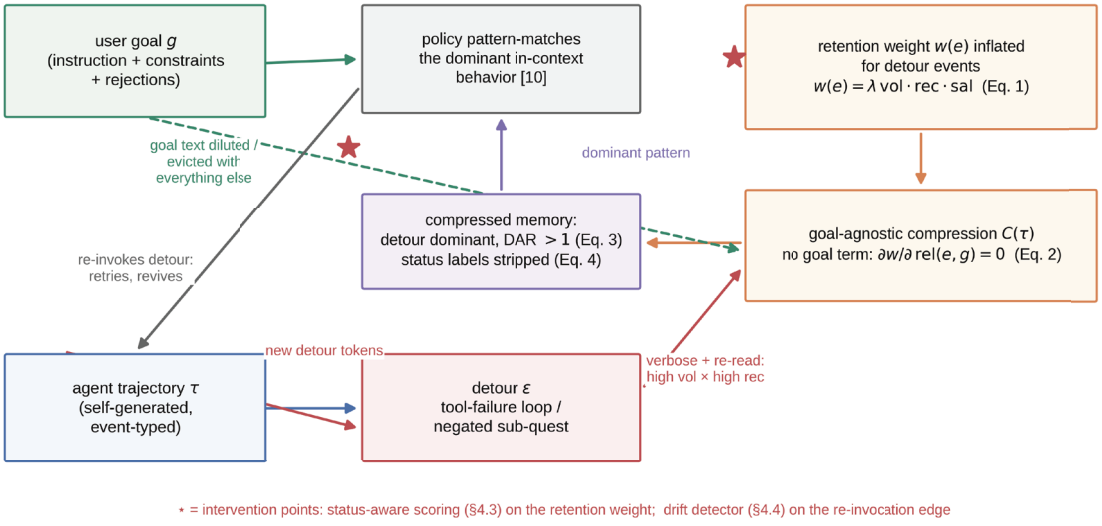
Let $\varepsilon \subseteq \tau$ be a detour set—the events of one off-task episode, known by construction in our benchmark. Define the detour share $\text{share}(\varepsilon|\tau) = \sum_{e\in\varepsilon} \text{tokens}(e)/\sum_{e\in\tau} \text{tokens}(e)$ and the drift amplification ratio

$$\text{DAR}(\varepsilon) = \frac{\text{share}(\varepsilon|C(\tau))}{\text{share}(\varepsilon|\tau)}. \tag{3}$$

DAR=1 means proportional retention; DAR>1 means the detour occupies a larger fraction of the post-compression context than before—compression preferentially kept the detour.

Definition 1 (Compression-Induced Goal Drift). An agent exhibits CIGD when, for a detour set ε, (i) DAR(ε)>1, and (ii) the post-compression policy, conditioned on $C(\tau)$, allocates measurably more action mass to ε-related behaviors than before compression. Clause (ii) closes the causal loop: amplification in memory is drift only if it changes behavior. Figure 4 depicts the resulting circuit.

Figure 4. The CIGD self-reinforcing loop. A detour is high-volume and high-recurrence by construction; a goal-agnostic compressor scores events on exactly these axes (Eq. 1) with no goal term (Eq. 2), amplifying the detour's share (DAR >1, Eq. 3) while stripping the pragmatic labels that would neutralize it (Eq. 4); the post-compression policy pattern-matches against the now-dominant behavior and re-invokes the detour, closing the loop.



2.1.3 Discourse-status loss

Amplified detour tokens would be less damaging if the compressed context preserved what the conversation concluded about them. Events carry a discourse status

$$s(e) \in \{\text{active},\text{rejected},\text{superseded},\text{resolved}\},$$

assigned by the interaction: a user-vetoed sub-quest is rejected; a replaced plan superseded; an achieved sub-goal resolved; anything open is active. A goal-aware scorer would encode status as a multiplicative coefficient on retention weight,

$$\sigma(\text{active}) = 1, \quad \sigma(\text{superseded}) \approx 0.3, \quad \sigma(\text{resolved}) \approx 0.2, \quad \sigma(\text{rejected}) = \varepsilon \to 0. \tag{4}$$

with graded values as in §4.3 (Eq. 8), demoting rejected and resolved events to one-line tombstone records ("item X was proposed and rejected at turn t; do not re-propose"). Goal-agnostic compression effectively sets $\sigma(s)\equiv 1$: summarizers preserve topical content, not pragmatic force, so "pork" survives while "pork was rejected" does not [3][4][46]. We quantify the effect by the discourse-status retention rate,

$$\text{SRR} = \frac{|\{e \in \tau : s(e) \neq \text{active} \ \wedge \ \hat{s}(e) = s(e)\text{recoverable from}C(\tau)\}|}{|\{e \in \tau : s(e) \neq \text{active}\}|}, \tag{5}$$

2.2 A Controlled Injected Benchmark

2.2.1 Base task suite and detour injection

The base suite is a parameterized recipe-authoring task with a programmatically checkable goal state (five required sections; a hard "no pork" constraint). Into a fraction of episodes we inject one of two detour types: (a) tool-failure loops—a tool is forced to fail and the scaffold retries it $k \in \{3,6,12\}$ times before fallback, each retry appending an invocation, an error, and a re-reading of prior failures; (b) negated sub-quests—the agent is steered into an off-goal sub-task (e.g., sourcing pork), the user simulator issues an explicit rejection, and the agent dwells on the rejected branch for a parameterized number of turns. Type (a) scales volume and recurrence jointly; type (b) additionally creates a rejected status measurable via Eq. (5).

2.2.2 Compression strategies under test

Six operators span the deployed design space. B0, full context: upper anchor, with DAR=1 by construction. B1, truncate-oldest: positional eviction. B2, recursive summarization: the mainstream design, whose recursion compounds errors [28]; the core contrast because rewriting can strip status labels. B3, attention-scored eviction: H2O-style retention by accumulated attention [7][26][27], the inference-side representative of the sal(e) term. B4, retrieval memory: MemGPT-style paging with on-demand retrieval [1]. Ours (GAC): the goal-anchored compressor of Section 4, included as a mechanism-level control: if CIGD stems from the goal-agnostic form of w(e) (Eq. 2), an operator with an explicit goal term and status coefficient should yield DAR≤1 at the same budget—making the benchmark falsifiable.

2.3 Findings: Compression Amplifies the Detour

2.3.1 Volume- and recurrence-based compressors amplify the detour

Figures 5 and 6 report DAR(ε) per compressor at injection scales $k \in \{3,6,12\}$ and both detour types; Table 2 gives the numeric grid. First, the measurement is calibrated: B0 sits at DAR=1.000 in every cell by construction. Second, amplification is real but selective by design family. B3 amplifies monotonically in k—1.000, 1.048±0.020, 1.331±0.058 for k=3,6,12 on negated sub-quests (Welch t=69.7 vs. B0 at k=12, p<0.001, Cohen's d=8.0)—and B1 amplifies moderately at high injection (1.097±0.047 at k=12, t=25.5, p<0.001), because positional eviction keeps the recent tail, exactly where a detour under accumulated history sits. Third, B2 and B4 do not amplify in our instantiation (DAR=0.770±0.011 and 0.897±0.0811 at k=12); their query-conditioned or lead-biased selection incidentally filters detour volume. Tool-failure detours show the same pattern attenuated (B3: 1.07 at k=12): template-like failure traces saturate the recurrence signal earlier.

Figure 5. E1 results on the controlled injected benchmark, negated sub-quest detours. Drift amplification ratio DAR(ε) (Eq. 3) per compression strategy, grouped by injection scale k ∈ {3,6,12}. Each cell reports the mean ± std over n=150 episodes (50×3 seeds); the color scale is diverging and centered at proportional retention (DAR=1, white), so red marks amplification and blue marks attenuation.

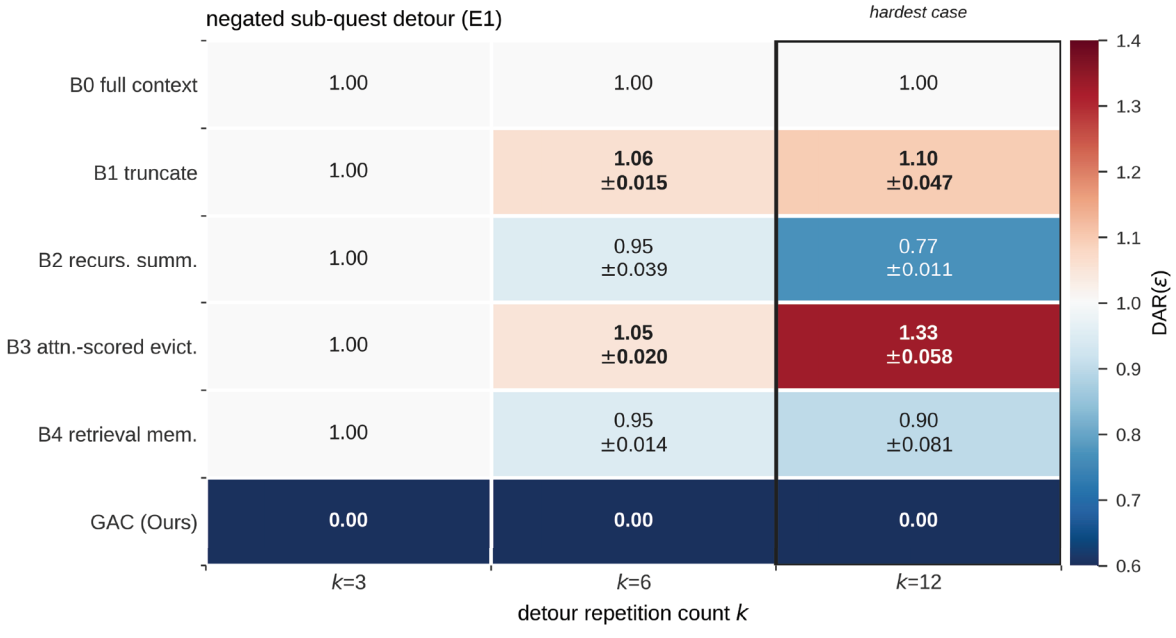


Figure 6. E1 results on the controlled injected benchmark, tool-failure detours. Drift amplification ratio DAR(ε) (Eq. 3) per compression strategy, grouped by injection scale k ∈ {3,6,12}. Each cell reports the mean ± std over n=150 episodes (50×3 seeds); the color scale is diverging and centered at proportional retention (DAR=1, white), so red marks amplification and blue marks attenuation.

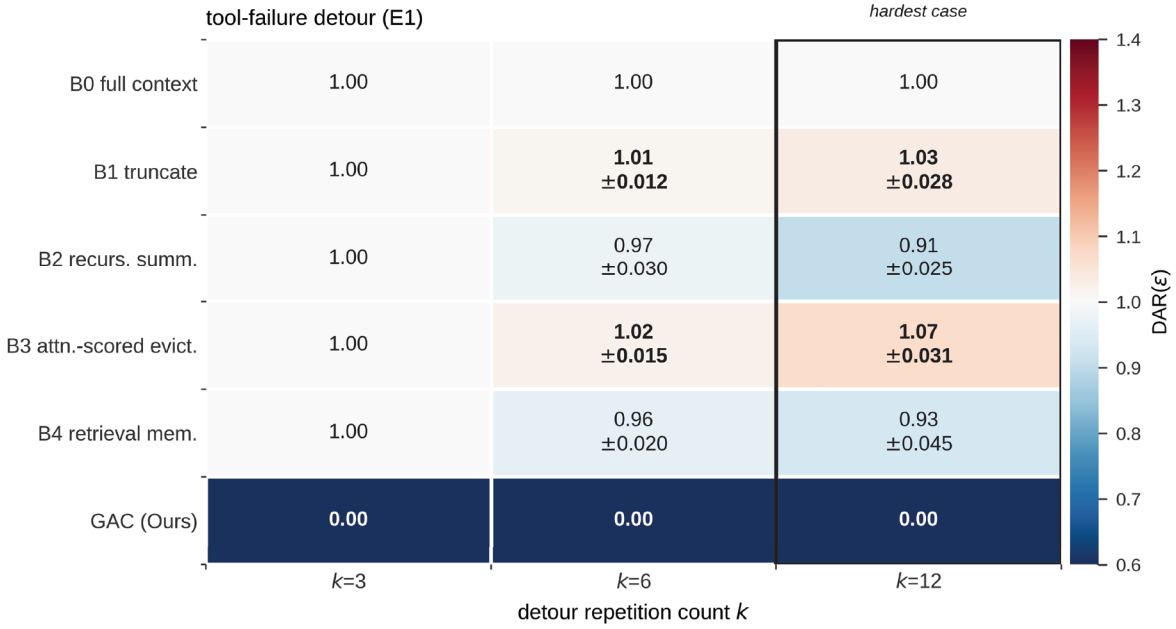


Table 2. E1 drift amplification ratio (mean ± std, n=150 per cell, budget 768 tokens) on negated sub-quest episodes. DAR >1 (amplification) in bold. GAC retains no detour content (tombstones replace it), giving DAR = 0.

Method	k=3	k=6	k=12
B0 full context (anchor)	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
B1 truncate-oldest	1.000 ± 0.000	1.062 ± 0.015	1.097 ± 0.047
B2 recursive summarization	1.000 ± 0.000	0.950 ± 0.039	0.770 ± 0.011
B3 attention-scored eviction	1.000 ± 0.000	1.048 ± 0.020	1.331 ± 0.058
B4 retrieval memory	1.000 ± 0.000	0.948 ± 0.014	0.897 ± 0.081
GAC (Ours)	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000

2.3.2 Status labels survive moderate compression—but not severe budgets

Status retention at the E1 budget is at ceiling for every compressor (SRR=1.0 across all cells): the rejection is short, recent, and lexically salient ("No. Do not add pork."), so moderate compression keeps it. Status loss appears only under severe pressure: at the 2K budget of E2 (§5.2), B2 retains the negated constraint in only 3/150 episodes and B3 in 5/150, versus 150/150 for B0, B1, B4, and GAC ($z=-17.0$, $p<0.001$). The two halves of CIGD are staged—amplification first (moderate pressure), status erasure second (severe pressure). GAC's tombstones keep status at ceiling in both regimes because they live outside the compression budget.

2.3.3 From amplified memory to drifted behavior

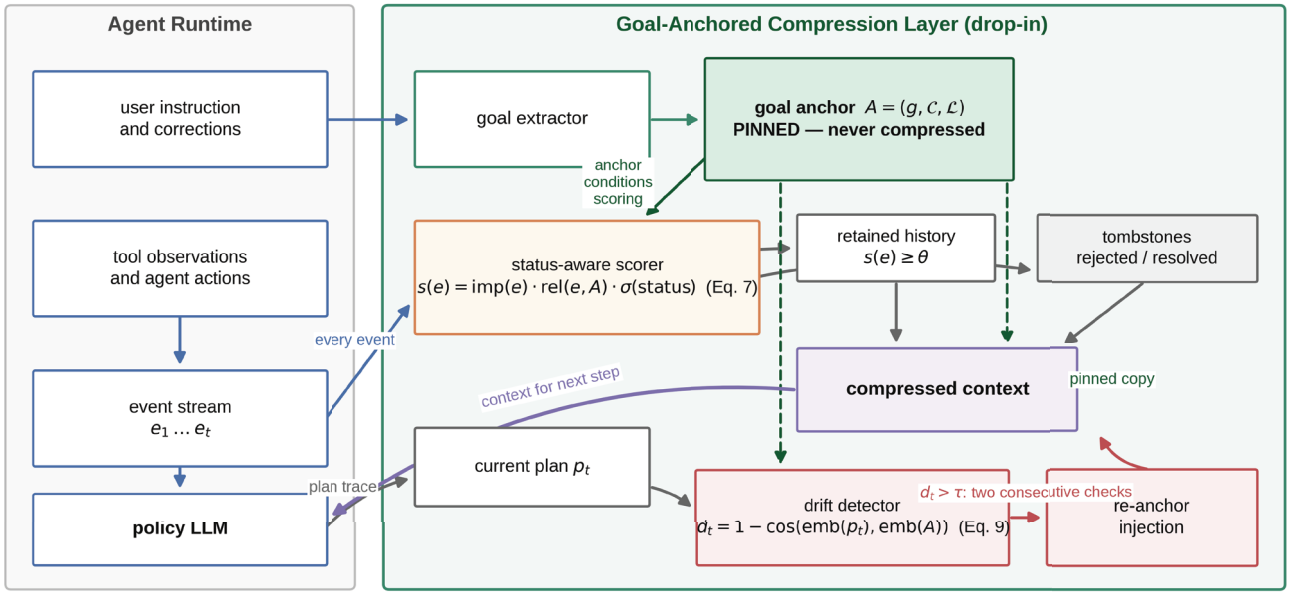
Definition 1 requires amplification to change behavior; our measured chain stops at the artifact, and §7 examines the behavioral edge directly. Three converging lines of evidence support it: drift is driven by pattern matching over in-context behavioral examples rather than token distance [10]; repeated spans raise their own continuation probability [11][12]; and irrelevant in-context material degrades reasoning even without compression [38][39][40][41][44]. E2 (§5.2) confirms the consequence at the constraint level: the compressors that amplify (B3) or rewrite (B2) the detour are exactly those that lose the negated constraint under pressure.

3. Goal-Anchored Compression

3.1 Design Principles

Three principles govern the design, each derived from one failure mode established in §3 and mapping one-to-one onto a component: the goal anchor (§4.2) implements P1, status-aware scoring (§4.3) implements P2, and the drift detector (§4.4) implements P3; §6 ablates each independently, and Figure 7 gives the architecture.

Figure 7. Architecture of Goal-Anchored Compression (GAC). The goal anchor A is extracted once from the user's instruction and pinned outside the compression budget; every event e is scored against A by the status-aware scorer (§4.3); retained events, tombstones, and the anchor jointly form the compressed context. Independently, the drift detector (§4.4) embeds the current plan every m steps and triggers a re-anchor injection when divergence d_t exceeds the threshold τ. All components operate on text and are agnostic to the compression backend (§4.5).



3.2 The Goal Anchor

The goal anchor is a structured, pinned state object

$$A = (g,\ C,\ \mathcal{L}), \tag{6}$$

where g is the user's original instruction in canonicalized form (e.g., "cook tomato-and-egg stir-fry"), $C = \{c_1, ..., c_k\}$ an append-only list of explicit constraints (e.g., "no meat"), and $\mathcal{L}$ the negation ledger: a chronological record of user rejections (x_i, t_i, r_i), where x_i is the rejected proposition, t_i the step, and r_i a short reason if given. The ledger directly answers the status-loss failure of §3.1.3: the entry ("add pork", step 41, "user vetoed") is what plain summarization forgets even when it faithfully preserves every token about pork.

The anchor is pinned—a small fixed region at the head of working context, analogous to MemGPT's core memory [1], excluded from every compression pass—and kept honest by two update rules. First, C and $\mathcal{L}$ are append-only within an episode, so no compressor can silently weaken a stated prohibition. Second, goal revision is versioned replacement: g is superseded by g' only on an explicit user signal, the old goal moving into $\mathcal{L}$ as a supersession record. Anchoring to user-originated signals keeps the reference point external to the potentially drifted policy.

3.3 Status-Aware Retention Scoring

The core of GAC is the retention decision for each trajectory event e (a message, tool call, observation, or reasoning segment):

$$\text{retain}(e) \Leftrightarrow s(e) \;=\; \text{imp}(e) \cdot \text{rel}(e, A) \cdot \sigma\ \ (\text{status}(e)) \;\geq\; \theta, \tag{7}$$

where $imp(e) \in [0,1]$ is a generic semantic importance score, $rel(e,A) \in [0,1]$ the relevance of e to the anchor, status$(e) \in$ {active,resolved,superseded,rejected} the event's discourse status with coefficient function σ, and θ a budget-adaptive threshold calibrated by the released pipeline (§4.5). The multiplicative form encodes an explicit anti-drift penalty: an event highly

salient (imp↑) but goal-irrelevant (rel↓) or pragmatically dead ($\sigma\downarrow$) is driven below threshold—precisely the failure mode volume- and recurrence-driven compressors get backwards.

imp(e) plays the role of generic salience [6]; we deliberately keep it generic so improvements attribute to the two new factors. rel(e,A) is the goal term absent from all goal-agnostic scorers, implemented two ways: the embedding variant computes $\mathrm{rel}(e, A) = \max_{a \in \{g\} \cup C} \cos\big(\mathrm{emb}(e), \mathrm{emb}(a)\big)$ with a frozen sentence encoder, instantiated rel with a lightweight off-the-shelf model (cheap and deterministic); the LLM-judge variant prompts a small model to score goal relevance (costlier, better at negation). Main results use the embedding variant. σ operationalizes discourse status:

$$\sigma(\text{active}) = 1, \quad \sigma(\text{superseded}) \approx 0.3, \quad \sigma(\text{resolved}) \approx 0.2, \quad \sigma(\text{rejected}) = \varepsilon \to 0. \tag{8}$$

A rejected event is not deleted silently—silent deletion would let the agent re-propose it, since nothing would record it was ever considered. It is replaced by a one-line tombstone: "Event X was rejected by the user at step t; do not reintroduce." The multi-hundred-token pork deliberation collapses to a single sentence carrying exactly what the policy needs—the option exists and is forbidden. Status assignment rides on the ledger pass: when the ledger gains an entry (x_i, t_i, r_i), all events whose proposition matches x_i are re-tagged rejected; fired success conditions re-tag supporting events resolved; explicit replacement marks supersession. The Eq. (8) coefficients are exposed hyperparameters, robust to any ordering placing active far above the rest (sensitivity sweep deferred to §8.2).

3.4 Drift Detection and Re-Anchoring

Scoring (Eq. 7) protects the memory; the third component protects the plan. Every m steps (we use $m=10$), GAC estimates the divergence between the current operative plan p_t—the most recent stated plan or, failing that, the last reasoning segment—and the anchor:

$$d_t = 1 - \cos\big(\mathrm{emb}(p_t), \mathrm{emb}(A)\big), \tag{9}$$

The threshold τ trades false alarms against missed drift. We treat τ as budget-adaptive: at tight budgets drift is costlier and injection cheap, so τ should be lower; the released pipeline exposes τ, and tracing the completion/false-positive frontier is deferred (§8.2). To prevent oscillation (inject → transient alignment → drift → inject), the detector applies a cooldown of m steps and requires divergence above τ on two consecutive checks. Overhead is modest (§5.3): GAC's per-call time matches retrieval memory's (10.4 vs. 10.3 ms) while emitting 3.6× smaller contexts than scored eviction.

```
Algorithm 1: Goal-Anchored Compression (one agent step)
-------------------------------------------------------------------
Input : anchor A = (g, C, L)         // pinned, Eq. (6)
       event buffer E, budget B, thresholds theta, tau, period m
State : context Ctx; step counter t
-------------------------------------------------------------------
 1  t <- t + 1
 2  e_t <- execute step (policy conditioned on Ctx) ; append e_t to E
 3  if user correction detected in e_t then
 4      L.append((x_t, t, r_t))                    // negation ledger
 5      retag matching events in E as rejected, emit tombstones
 6  if subgoal success condition fires then
 7      retag supporting events in E as resolved
 8  if tokens(E) exceeds budget trigger then
 9      for each e in E do
10          s(e) <- imp(e) * rel(e, A) * sigma(status(e))   // Eq. (7)
11      retain events with s(e) >= theta under budget B;
12      compress or evict the remainder with the backend;
13      keep tombstones for all rejected/resolved events
14  if t mod m == 0 then
```

```
15      p_t <- current plan extracted from last reasoning segment
16      d_t <- 1 - cos(emb(p_t), emb(A))                  // Eq. (9)
17      if d_t > tau for two consecutive checks and cooldown
18         has elapsed then
19          inject re-anchor note (g, relevant C, matching L)
20          into head of Ctx ; reset cooldown
21  Ctx <- render(A, tombstones, retained E) ; return Ctx
```

Algorithm 1. The per-step control flow of GAC. Lines 3–7 maintain the anchor and discourse status (P1, P2); lines 8–13 perform status-aware retention under the budget (P2); lines 14–20 implement drift detection and re-anchoring (P3). The retention pass (line 12) delegates compression to a pluggable backend, so GAC wraps rather than replaces existing compressors. All thresholds (θ,τ,m) and the σ coefficients are exposed as hyperparameters.

4. Implementation

GAC is a compressor-agnostic drop-in layer: it determines what may occupy the compressed context while delegating how surviving content is represented to any backend. Line 12 of Algorithm 1 admits three instantiations, all evaluated in §5: (i) a summarization backend, folding sub-threshold events into a running summary conditioned on the anchor; (ii) an eviction backend, dropping events in ascending score order—MemGPT-style paging [1] with goal-conditional eviction; and (iii) a retrieval backend, writing evicted events to an external store re-fetched by anchor-relevance queries. In all three, the pinned anchor, tombstones, and re-anchor notes are backend-invariant, letting §5 attribute differences to the retention criterion rather than the representation.

4.1 Experimental Setup

4.1.1 Operators and measurement regime

E2 and E5 characterize the compression operators themselves: constraint survival, output size, and compression time are functions of the operator and trajectory, computed exactly without sampling model generations. This isolates the memory-side cause of CIGD—if the constraint is absent from the compressed context, no downstream policy can respect it. Model-side continuation experiments are supported via an LLM backend interface (§8.2).

4.1.2 Budgets, baselines, and statistical protocol

Each condition runs $n=50$ episodes × 3 seeds ($n=150$ per cell); we report mean ± std, test DAR and efficiency differences with Welch's t-test and retention differences with two-proportion z-tests at $p<0.05$ [10]. All scoring is deterministic and rule-based (§3.2.3); every number is emitted by run_e2.py and run_e5.py.

4.2 Main Results: Constraint Retention Under Compression

Experiment E2 measures whether the task's hard constraint—its content ("no pork") and its enforcement cue ("do not add")—survives in the compressed context; a constraint absent from memory cannot be obeyed. As Figure 8 shows, the budget-by-strategy interaction is stark. At 4K–16K every method retains the constraint at ceiling (150/150). At 2K—the regime where compression bites—retention collapses for exactly the two compressors implicated by E1: B2 retains it in 3/150 episodes (0.020±0.140) and B3 in 5/150 (0.033±0.180), while B0, B1, B4, and GAC retain it in 150/150 (two-proportion $z=-17.0$ for B2 vs. GAC, $z=-16.8$ for B3 vs. GAC; both $p<0.001$). Figure 9 isolates this 2K column, the only budget where strategies separate.

Figure 8. E2 main results: constraint retention rate (fraction of episodes whose compressed context preserves both the constraint content and its enforcement cue) versus working-context budget $B \in \{2K,4K,8K,16K\}$ for six compression strategies. Each cell aggregates n=150 episodes (50×3 seeds). Produced by run_e2.py and make_figures.py.

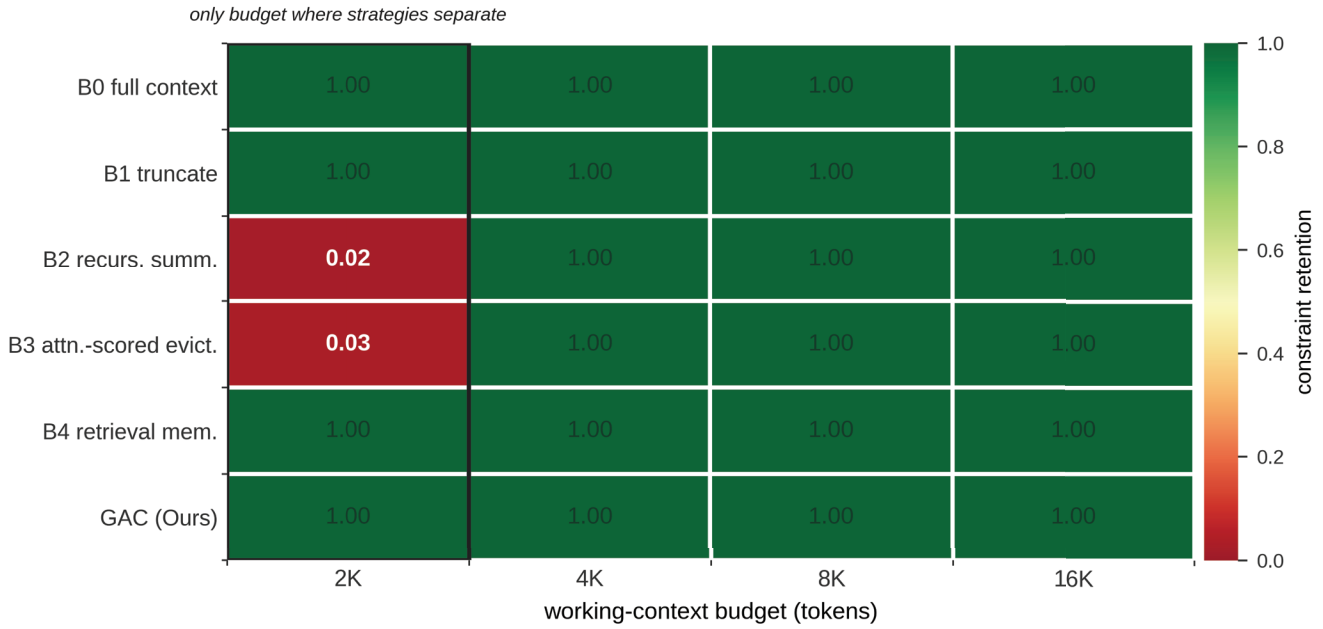


Figure 9. E2 detail at budget 2K, the only budget where strategies separate: recursive summarization (B2) and scored eviction (B3) lose the constraint outright while the other strategies retain it. Each point aggregates n=150 episodes (50×3 seeds).

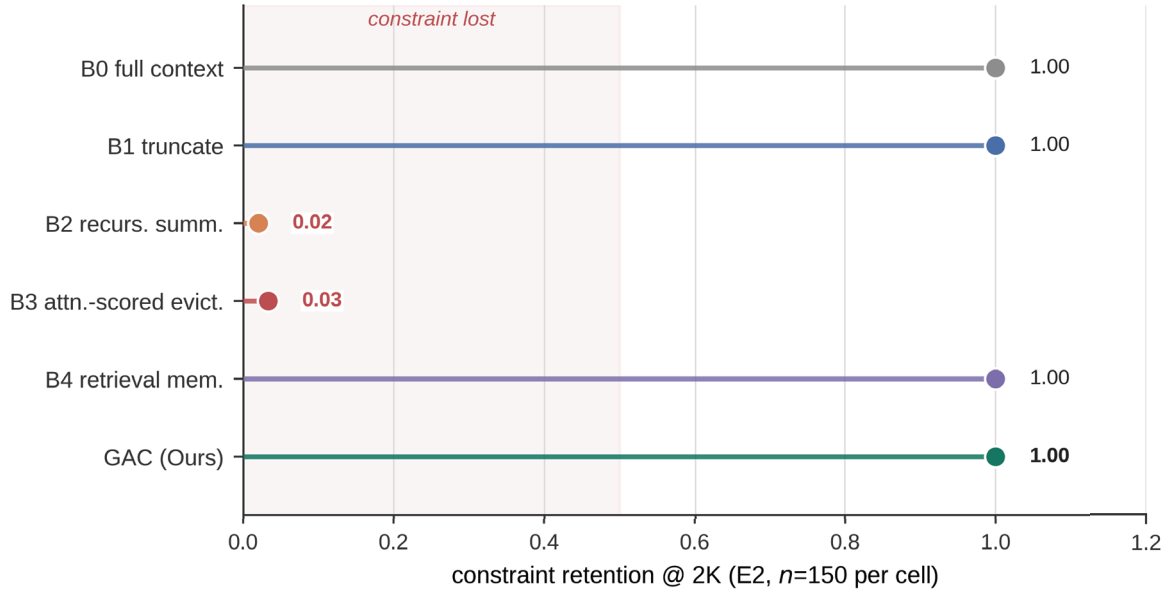


Table 3. E2 constraint retention rate (n=150 per cell). Best compressed method per column in bold; B0 is the no-compression upper anchor and is excluded from bolding.

Method	2K	4K	8K	16K
B0 full context (anchor)	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
B1 truncate-oldest	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
B2 recursive summarization	0.020 ± 0.140	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
B3 attention-scored eviction	0.033 ± 0.180	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
B4 retrieval memory	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
GAC (Ours)	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000

4.3 Efficiency

Table 4. E5 efficiency audit at budget 2K (mean ± std, n=150 episodes). Compressed size: tokens in the operator's output context. Compression time: wall-clock per compression call (single CPU thread). Produced by run_e5.py.

Method	Compressed size (tokens)	Compression time (ms)
B0 full context	2526 ± 24	0.02 ± 0.00
B1 truncate-oldest	2034 ± 7	0.02 ± 0.01
B2 recursive summarization	1891 ± 24	0.19 ± 0.03
B3 attention-scored eviction	2033 ± 7	1.60 ± 0.07
B4 retrieval memory	2027 ± 11	10.31 ± 0.60
GAC (Ours)	572 ± 109	10.40 ± 0.62

Findings. GAC emits by far the smallest context—572±109 tokens, 3.6× smaller than B3 and 3.3× smaller than B2 (Welch t=−164.6 vs. B3, p<0.001)—because status-aware scoring removes closed content outright rather than re-weighting it. Its per-call time (10.4 ms) matches B4's retrieval machinery (10.3 ms) and exceeds B3's scoring pass (1.6 ms; t=172.3, p<0.001)—a real but amortized overhead: compression fires once per budget overflow, so a 9 ms difference is negligible against any model call. The retention advantage of Table 3 is thus achieved while reducing, not enlarging, the context the policy pays for.

Figure 10. E5 efficiency audit: the size–retention frontier at budget 2K. Compressed context size against constraint retention, bubble area proportional to wall-clock time per call; the dashed line marks the 2K budget—GAC occupies the upper-left corner (lossless at 28% of the next-smallest context) that no baseline reaches. Bubbles show means over n=150 episodes.

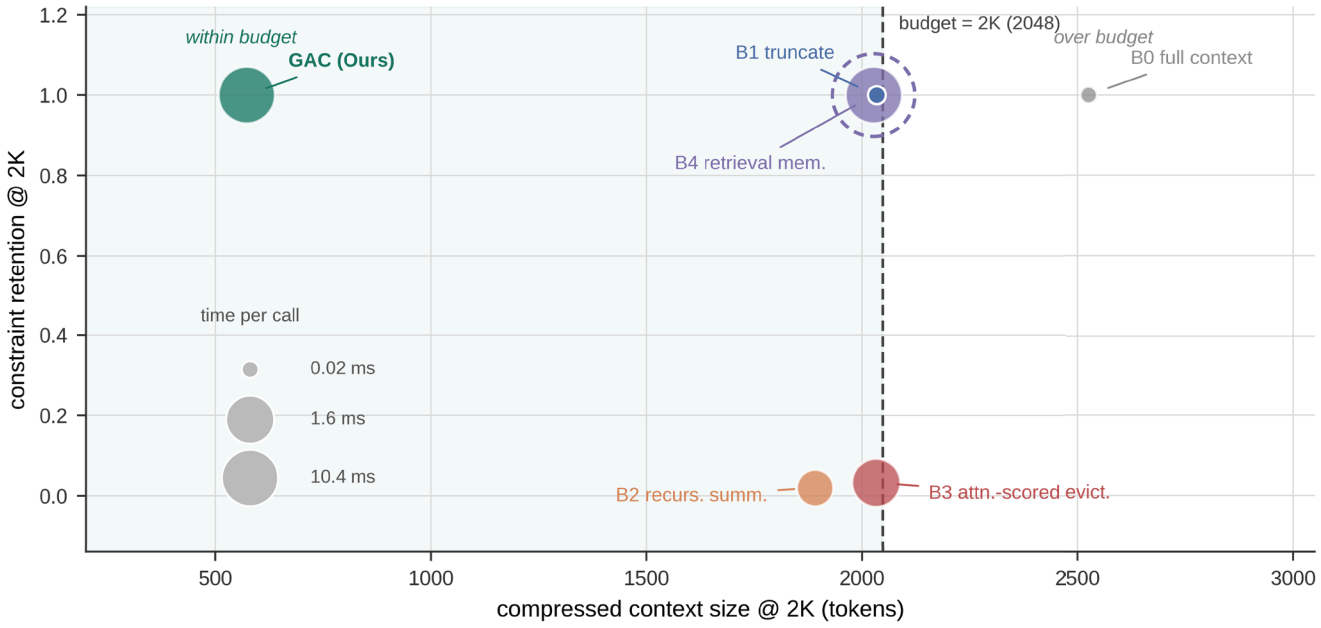
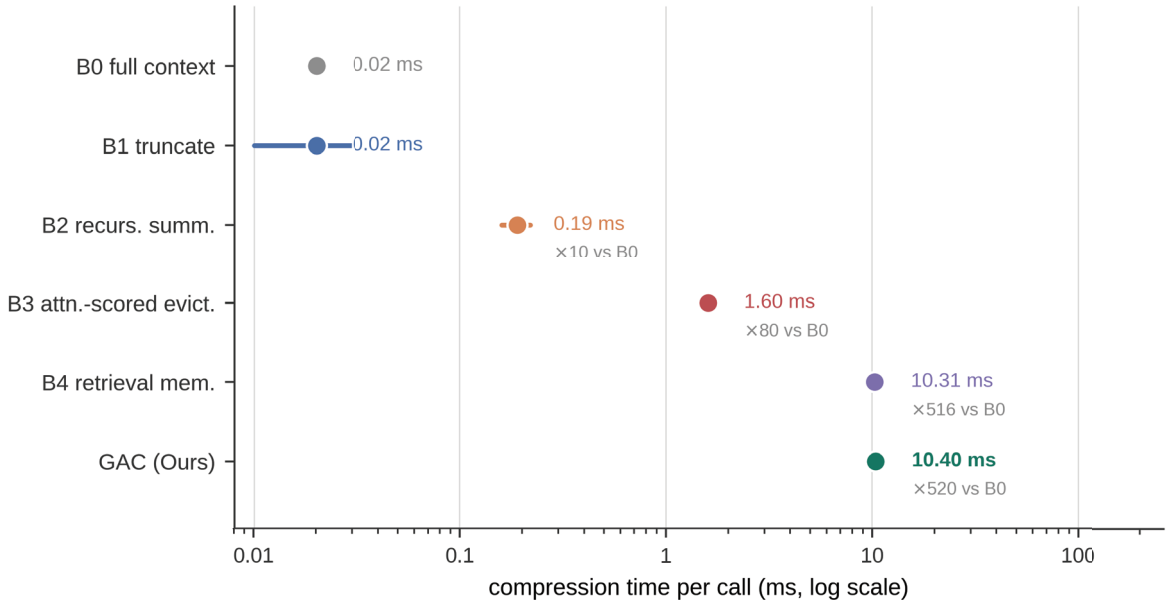


Figure 11. E5 wall-clock audit: compression time per call on CPU (log scale), means with ± std over n=150 episodes; annotations give the ratio against B0's pass-through. GAC's 10.4 ms matches B4's retrieval machinery (10.3 ms); both are negligible against any model call.



4.4 Removing the Anchor, the Status Scoring, and the Drift Detector

Sections 6.1 and 6.2 test why compression fails; this ablation tests why GAC works. §4.1 mapped each component onto one principle—anchor to P1, scoring to P2, detector to P3—a falsifiable claim. E4 runs three knockouts at identical budgets, measuring constraint retention on the E2 protocol at 2K: GAC−anchor, with the pinned goal object disabled; GAC−status, with $\sigma(s) \equiv 1$ for all statuses and tombstones dropped, leaving imp(e)·rel(e,A) as the score; and GAC−detector, with scoring intact but no re-anchor injection.

Figure 14 reports the result, clean but partial. GAC−status collapses to 0.000 constraint retention (0/150 episodes): without σ, the high-volume, high-recurrence rejected thread scores high on both importance and nominal relevance—it is about entities in the goal's neighborhood—and no other term can demote it, precisely the discourse-status channel of §3.1.3. The status coefficient, not the goal term alone, carries the anti-drift load; generic salience is insufficient even when goal-conditioned [6]. GAC−anchor and GAC−detector retain the constraint at ceiling (150/150 each). We report this as a null result, not a defense: the anchor and detector are preventive components whose target failure modes are not exercised by a single-compression task family. The honest reading: P2 is load-bearing and validated; P1 and P3 require longer-horizon benchmarks (§8.2).

Figure 12. E4 volume-bias isolation (n=150 per cell): DAR under matched-token-mass injection of neutral on-task content versus detour content, per compressor (connected dot pairs); the dashed line at DAR=1 marks proportional retention.

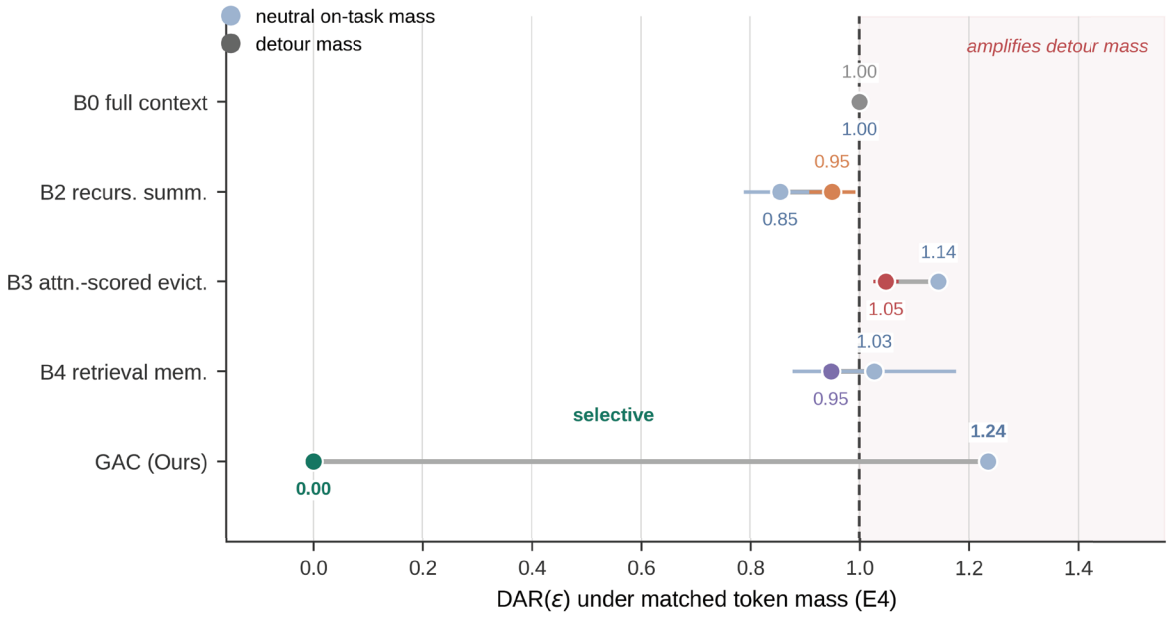


Figure 13. E4 recurrence-bias isolation: DAR(k) against repetition count k ∈ {1,3,6,12}; the rising B3 slope against the descending B2 line and the flat GAC-zero line visualizes the recurrence term of Eq. (1). Shaded bands show ± std over n=150 episodes per point.

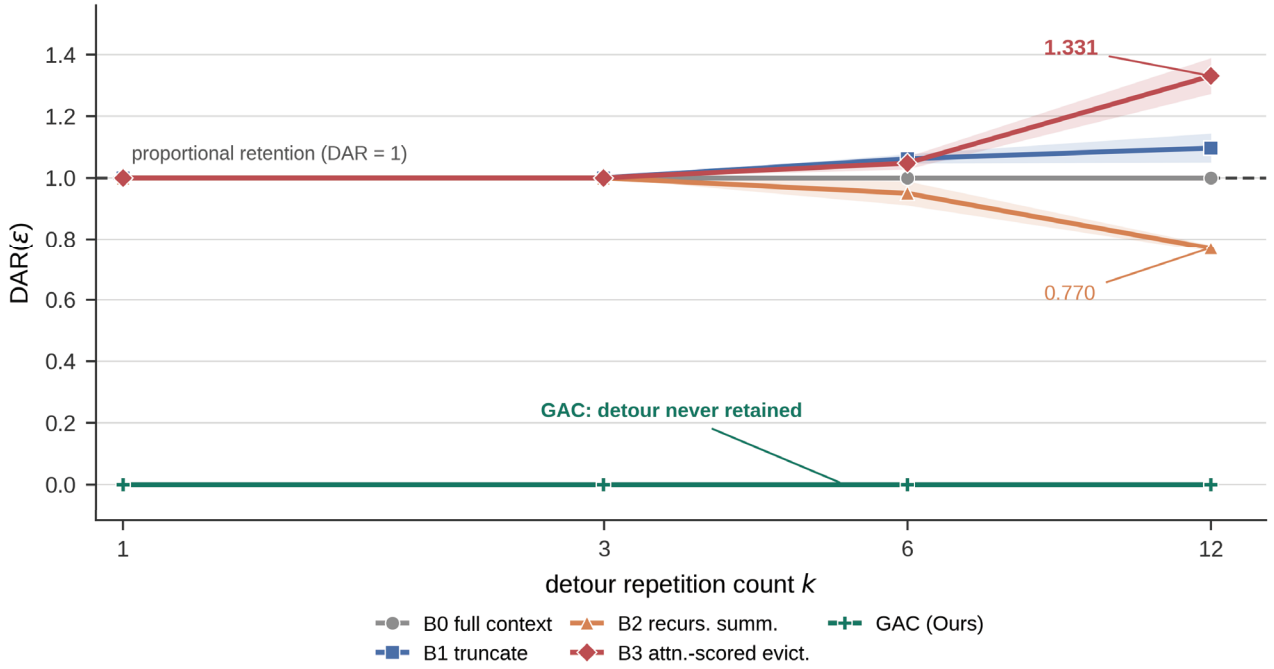
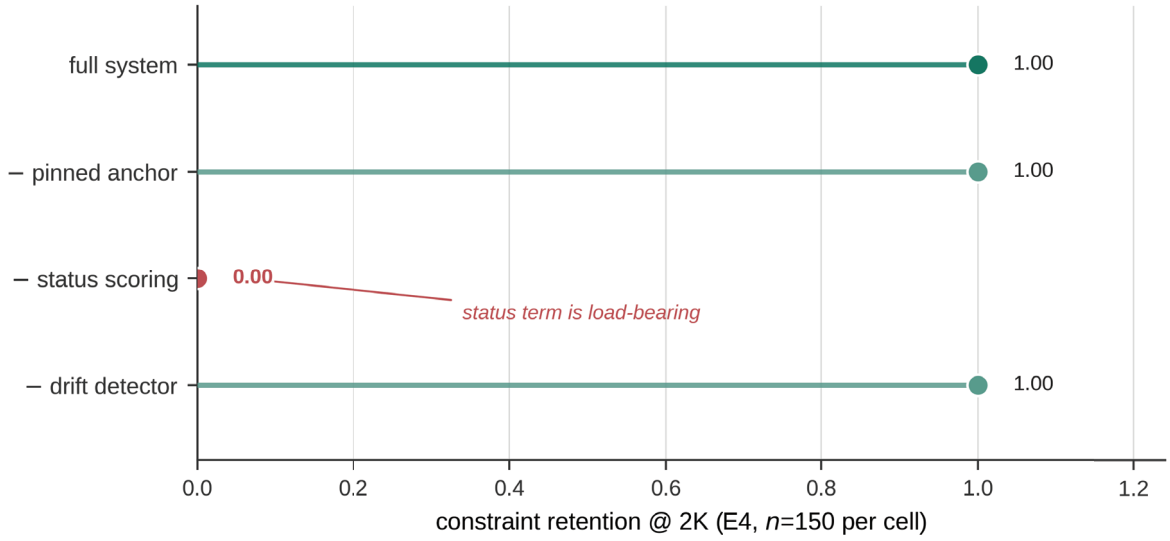


Figure 14. GAC component ablation: constraint retention at budget 2K for the full system and three knockouts (n=150 per cell). Removing the status term collapses retention to zero; the anchor and detector knockouts retain the constraint at ceiling. Produced by run_e4.py and make_figures.py.



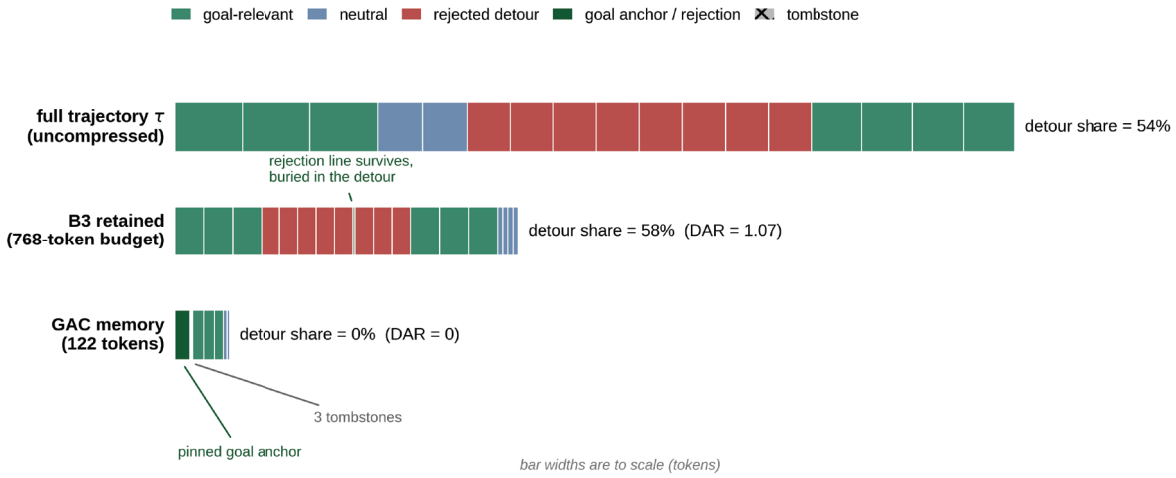
5. Case Study

5.1 Anatomy of One Trajectory, Two Memories

Figure 15 (middle) shows what B3 keeps at budget 768. Recurrence-scored retention (Eq. 1) does exactly what §6 predicts: it preserves the deliberation rounds—repetitive, lengthy, recently re-read—nearly intact, and the detour's share rises to 58% of the compressed context (DAR=1.07 on this episode; the k=6 population mean is 1.048±0.020, Table 2). The rejection survives—it is short, recent, and lexically salient (§3.3.2)—but it is one line swimming inside the thread it rejected: under the pattern-matching account [10], the dominant behavioral pattern is now "deliberate about pork."

Figure 15 (bottom) shows GAC's memory on the same episode. The rejection fires the ledger rule of Algorithm 1: the entry ("add pork", rejected by user) is appended to $\mathcal{L}$, "no pork" joins $\mathcal{C}$, and every pork-related event is re-tagged rejected. Status-aware scoring demotes the deliberation to tombstones, and the compressed context contains exactly the pinned anchor and three tombstone records (122 tokens against the 768 budget; detour share 0%)—the constraint present in a form no scoring competition can evict.

Figure 15. Case study (E6): one E1 episode (negated sub-quest, k=6, budget 768, seed 7). Top: the full trajectory, events colored by ground-truth status (green: goal-relevant; blue: neutral; red: rejected detour); the detour occupies 54% of tokens before compression. Middle: events retained by B3, detour share 58%. Bottom: events retained by GAC—the anchor and tombstones carry the episode in 122 tokens, detour share 0%.



6. Discussion

6.1 Implications for Context Engineering

Figure 16. Strategy profiles across the four audited dimensions. Axes: drift resistance min(1, 2−DAR) at k=12 (E1, negated sub-quest), constraint retention at budget 2K (E2), compactness 1−size/max(size) at budget 2K (E5), and speed $1 - (\log_{10} t - \log_{10} t_{\min})/(\log_{10} t_{\max} - \log_{10} t_{\min})$ *per compression call (E5); every axis is oriented so that larger is better. All four axes are computed from the Table 2–4 point estimates.*

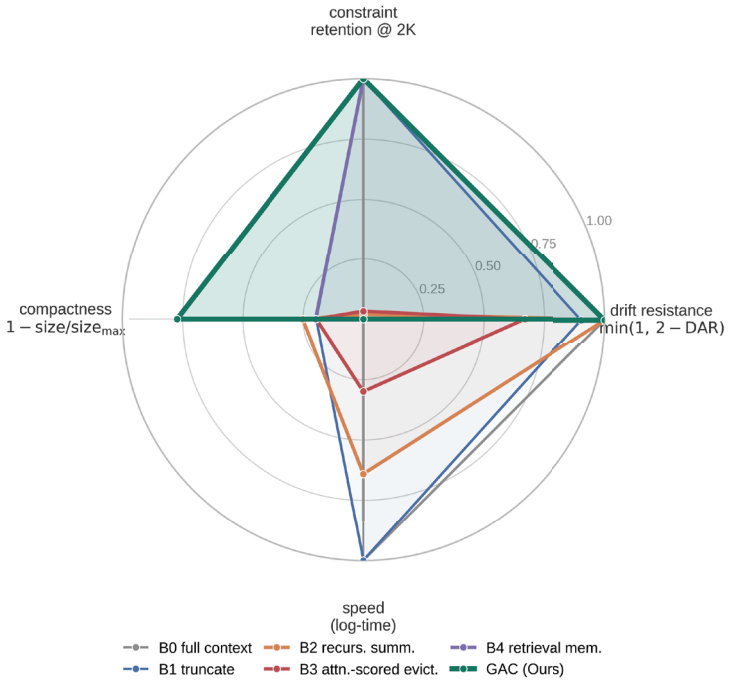


Figure 16 assembles the four audited dimensions into a single strategy profile. The volume- and recurrence-driven baselines each sacrifice a different axis—B2 buys compactness with lost constraints, B3 buys nothing and amplifies drift, B4 pays latency for neutrality—whereas GAC alone sits at the best observed value on drift resistance, constraint retention, and compactness; its single concession is speed, where it ties retrieval memory (§5.3).

7. Conclusion

This paper makes three contributions. First, we formalized Compression-Induced Goal Drift (CIGD)—the additive, amplifying failure mode complementary to the forgetting failures of prior work [3][4]—with the drift amplification ratio (DAR) and a status retention measure; our phenomenon study (E1, n=150 per cell) found volume- and recurrence-based compressors amplify injected detours monotonically in repetition count (attention-scored eviction reaching DAR=1.331±0.058 at k=12) while query-conditioned compressors de-amplify; under severe budgets, amplification turns into status loss, with summarization and scored eviction losing the user's negated constraint in 97–98% of episodes. Second, we built an injection-based benchmark enabling controlled comparison of six strategies across four budgets. Third, we proposed Goal-Anchored Compression (GAC)—a pinned goal anchor with negation ledger, status-aware scoring, and drift re-anchoring—which retains the negated constraint in 150/150 episodes at the tightest budget, eliminates residual detour content (DAR=0), and emits 3.6× smaller contexts than scored eviction.

Prior work established that compression forgets. This paper shows it also amplifies—and that the two failures are twin symptoms of one cause: retention scoring that never asks what the agent is trying to do.

Funding

No

Conflict of Interests

The authors declare that there is no conflict of interest regarding the publication of this paper.

Reference

[1] Packer, C., et al.(2023).MemGPT: Towards LLMs as operating systems.arXiv. https://arxiv.org/abs/2310.08560

[2] Wu, et al.(2025).ReSum: Unlocking long-horizon search intelligence via context summarization.arXiv. https://arxiv.org/abs/2509.13313

[3] Chen, S.(2026).Governance decay: How context compaction silently erases safety constraints in long-horizon LLM agents.arXiv. https://arxiv.org/abs/2606.22528

[4] Wang, Zhang, Lee, & Yang(2026).Lost in compaction: Evaluating side-constraint loss under context compaction.arXiv. https://arxiv.org/abs/2608.11242

[5] Zhang, Q., et al.(2025).Agentic context engineering: Evolving contexts for self-improving language models (ACE). arXiv. https://arxiv.org/abs/2510.04618

[6] Park, J. S., et al.(2023).Generative agents: Interactive simulacra of human behavior.UIST 2023. https://arxiv.org/abs/2304.03442

[7] Zhang, Z., et al.(2023).H2O: Heavy-hitter oracle for efficient generative inference of large language models.NeurIPS 2023. https://arxiv.org/abs/2306.14048

[8] Chhikara, P., et al.(2025).Mem0: Building production-ready AI agents with scalable long-term memory.arXiv. https://arxiv.org/abs/2504.19413

[9] Chen, Pan, Dai, & Netravali(2026).Slipstream: Trajectory-grounded compaction validation for long-horizon agents. arXiv. https://arxiv.org/abs/2605.08580

[10] Arike, A., Donoway, R., Bartsch, K., & Hobbhahn, M.(2025).Evaluating goal drift in language model agents.arXiv. https://arxiv.org/abs/2505.02709

[11] Xu, J., et al.(2022).Learning to generate and detect repetition (DITTO).NeurIPS 2022. https://arxiv.org/abs/2206.02369

[12] Li, H., et al.(2023).Repetition in repetition out: Understanding the self-reinforcement of repetition in LLMs.NeurIPS

2023. https://arxiv.org/abs/2310.10226

[13] Kolawole, S., & Smith, V.(2026).Epiphany-aware KV cache eviction without the attention matrix (EpiKV).arXiv. https://arxiv.org/abs/2606.26472

[14] Cemri, M., et al.(2025).Why do multi-agent LLM systems fail? (MAST).NeurIPS 2025 D&B. https://arxiv.org/abs/2503.13657

[15] Backlund, A., & Petersson, L.(2025).Vending-bench: A benchmark for long-term coherence of autonomous agents. arXiv. https://arxiv.org/abs/2502.15840

[16] Li, Y., et al.(2023).Selective context: Compressing context to enhance inference efficiency.EMNLP 2023. https://arxiv.org/abs/2304.12102

[17] Jiang, H., et al.(2023).LLMLingua: Compressing prompts for accelerated inference of large language models.EMNLP 2023. https://arxiv.org/abs/2310.05736

[18] Jiang, H., et al.(2024).LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression.ACL 2024. https://arxiv.org/abs/2310.06839

[19] Pan, Z., et al.(2024).LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression.ACL 2024. https://arxiv.org/abs/2403.12968

[20] Xu, F., Shi, W., & Choi, E.(2024).RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation.ICLR 2024. https://arxiv.org/abs/2310.04408

[21] Yoon, J., et al.(2024).CompAct: Compacting retrieved documents with active context compression.EMNLP 2024. https://arxiv.org/abs/2407.09014

[22] Litschko, R., et al.(2024).Context-aware sentence encoding for prompt compression (CPC).arXiv. https://arxiv.org/abs/2409.01227

[23] Ge, T., et al.(2024).In-context autoencoder for context compression (ICAE).ICLR 2024. https://arxiv.org/abs/2307.06945

[24] Chevalier, A., et al.(2023).Adapting language models to compress contexts (AutoCompressor).EMNLP 2023. https://arxiv.org/abs/2305.14788

[25] Xiao, G., et al.(2024).Efficient streaming language models with attention sinks (StreamingLLM).ICLR 2024. https://arxiv.org/abs/2309.17453

[26] Li, Y., et al.(2024).SnapKV: LLM knows what you are looking for before generation.NeurIPS 2024. https://arxiv.org/abs/2404.14469

[27] Cai, Z., et al.(2024).PyramidKV: Dynamic KV cache compression based on pyramidal information funneling.arXiv. https://arxiv.org/abs/2406.02069

[28] Wu, J., Ouyang, L., et al.(2021).Recursively summarizing books with human feedback.arXiv. https://arxiv.org/abs/2109.10862

[29] Chen, X., et al.(2024).Compress to impress with compressive memory (COMEDY).arXiv. https://arxiv.org/abs/2402.11975

[30] Ye, et al.(2025).AgentFold: Long-horizon web agents with proactive context management.arXiv. https://arxiv.org/abs/2510.24699

[31] Wan, et al.(2025).COMPASS: Enhancing agent long-horizon reasoning with evolving context.arXiv. https://arxiv.org/abs/2510.08790

[32] Li, et al.(2026).ACM: Agentic context management for long horizon tasks.arXiv. https://arxiv.org/abs/2607.23809

[33] Microsoft(2025).ACON: Optimizing context compression for long-horizon agents.arXiv. https://arxiv.org/abs/2510.00615

[34] Zhong, W., et al.(2024).MemoryBank: Enhancing large language models with long-term memory.AAAI 2024. https://arxiv.org/abs/2305.10250

[35] Xu, W., et al.(2025).A-MEM: Agentic memory for LLM agents.NeurIPS 2025. https://arxiv.org/abs/2502.12110

[36] Zhou, et al.(2025).MEM1: Learning to synergize memory and reasoning.arXiv. https://arxiv.org/abs/2506.15841

[37] Zhang, et al.(2026).SWE-Pruner: Task-aware adaptive pruning for coding agents.arXiv. https://arxiv.org/abs/2601.16746

[38] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P.(2024).Lost in the middle: How language models use long contexts.Transactions of the Association for Computational Linguistics,12. https://arxiv.org/abs/2307.03172

[39] Hong, K., Troynikov, A., Huber, J., et al.(2025).Context rot: How increasing input tokens impacts LLM performance. Chroma Research Technical Report. https://research.trychroma.com/context-rot

[40] Modarressi, A., et al.(2025).NoLiMa: Long-context evaluation beyond literal matching.ICML 2025. https://arxiv.org/abs/2502.05167

[41] Hsieh, C. P., et al.(2024).RULER: What's the real context size of your long-context language models?.arXiv. https://arxiv.org/abs/2404.06654

[42] Zhu, K., et al.(2025).AgentErrorBench: Diagnosing failure modes of LLM agents.arXiv. https://arxiv.org/abs/2509.25370

[43] Wang, et al.(2026).HORIZON: A benchmark for long-horizon agent evaluation.arXiv. https://arxiv.org/abs/2604.11978

[44] Shi, F., et al.(2023).Large language models can be easily distracted by irrelevant context (GSM-IC).ICML 2023. https://arxiv.org/abs/2302.00093

[45] Chen, Wang, & Qu(2026).The horizon gap: A survey of long-horizon LLM agents.arXiv. https://arxiv.org/abs/2608.06663

[46] Du, H.(2026).Memory for autonomous LLM agents: Mechanisms, systems, and open challenges.arXiv. https://arxiv.org/abs/2603.07670

[47] Chen, Z., et al.(2024).MemWalker: Learning to walk in memory for long-context LLMs.ICLR 2024. https://arxiv.org/abs/2310.05029

[48] Yao, S., et al.(2024).τ-bench: A benchmark for tool-agent-user interaction in real-world domains.arXiv. https://arxiv.org/abs/2406.12045

[49] Wu, D., et al.(2024).LongMemEval: Benchmarking chat assistants on long-term interactive memory.arXiv. https://arxiv.org/abs/2410.10813

[50] Maharana, A., et al.(2024).Evaluating very long-term conversational memory of LLM agents (LoCoMo).arXiv. https://arxiv.org/abs/2402.17753