A MEMORY YOU CAN'T RECALL IS NO MEMORY AT ALL: MEMENTO FOR QUERY-AGNOSTIC KV CACHE COMPRESSION IN MULTI-TURN INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-turn agentic inference requires a bounded KV cache to support information needs that change across user requests and tool interactions. Repeatedly evicting entries according to the current query can specialize memory to the task at hand: once discarded, evidence needed by a later turn cannot be recovered by rescoring the surviving cache. We argue that useful memory requires both availability and accessibility: a memory the model cannot recall cannot support its response. We propose **MEMENTO**, a training-free streaming compression method that lets the model reread: it re-feeds each completed context chunk, and the attention of the reread supplies query-agnostic retention scores, using the content itself to generate retrieval cues. MEMENTO interleaves rereading with subsequent decoding in isolated batch rows of the same forward pass, amortizing scoring latency while maintaining a persistent cache across turns. A geometric analysis interprets reread queries as witnesses of retrieval cells and bounds local attention error on the regions they cover. Under explicit coverage and value-separation conditions, the same analysis yields a lower expected future-query error than a cache specialized to the current query at the same retention budget.

1 INTRODUCTION

Cognitive psychology draws a line that everyday intuition misses: a memory being *stored* is not the same as a memory being *retrievable*. In their classic study of word recall, Tulving & Pearlstone (1966) separate the *availability* of a trace, its continued presence in memory, from its *accessibility*, whether a cue can bring it back: subjects failed to recall words they demonstrably still held, then recovered them the moment the right category cue was supplied. What governs behavior at a given moment is not what is stored but what a cue can reach, and **a memory the model cannot recall is, for the purposes of the present response, no memory at all.**

A transformer's key–value (KV) cache is exactly such a store, and attention is its cue-driven retrieval (we say *cue* for the text that prompts a retrieval and *query* for the vector through which attention carries it out). Each cached entry is a retained trace, and at each step a query decides, through the attention softmax, which traces are reached and which stay dormant, by the same normalization with which models of recall sample a trace (Raaijmakers & Shiffrin, 1981). Tulving's distinction therefore transfers verbatim: an entry is *available* when it sits in the cache, but *accessible* only when some future query makes attention fetch it. Because that fetch is normalized over every entry, **accessibility is a relation between an entry and the rest of memory, not a property the entry carries on its own**. An entry can be important and still be unreachable, with nothing about itself to say which.

This gap turns decisive the moment memory must be bounded. Multi-turn and agentic workloads grow the KV cache without limit as reasoning, tool calls, and observations accumulate (Anthropic, 2025; OpenAI, 2025; Google, 2025; Moonshot AI, 2026), forcing the inference stack to evict entries *before* it knows what any future turn will ask. Eviction is thus a blind bet on future accessibility, and existing compressors place that bet in one of two ways, neither of which measures it. Query-dependent methods score each entry by the attention from the *current* request (Li et al., 2024; Oren et al., 2024; Tang et al., 2024; Behnam et al., 2025; Le et al., 2026), which specializes the surviving cache to the cue already seen; when a later turn arrives with a different need, the entries it must reach may be

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

exactly the ones an earlier request judged unimportant. Query-agnostic methods drop the query and rank a trace by its content, position, or geometry (Xiao et al., 2024b; Park et al., 2025; Devoto et al., 2025), which avoids overfitting to a single request but still scores what an entry *contains* rather than whether attention can reach it. Both, in Tulving's terms, measure availability, present-cue reachability in the first case and raw content in the second, and neither asks the question the distinction makes operative:

When a future cue arrives, will attention actually reach this entry?

Keeping an important but unreachable entry wastes a scarce budget, and discarding a reachable one is, because eviction is irreversible, a permanent loss that every later turn inherits.

To measure accessibility, one must run retrieval. Whether an entry is accessible depends on what attention fetches, so no statistic computed outside attention can measure it; the only faithful test is to issue a cue and see where attention goes. The cue that matters, however, belongs to a turn that has not arrived yet. Human writers face the same problem at every sentence, and they handle it by rereading: they review what they just wrote to decide what to carry into the next sentence (Hayes & Flower, 1980; Kellogg, 1996). This works because what comes next draws on what came just before, and an agentic session behaves the same way: its next turns query the cache with content like its recent chunks. We therefore let the model reread. Re-feeding a completed chunk turns the queries that wrote it into queries that read it, and the entries they reach are the ones a future turn is likely to recall.

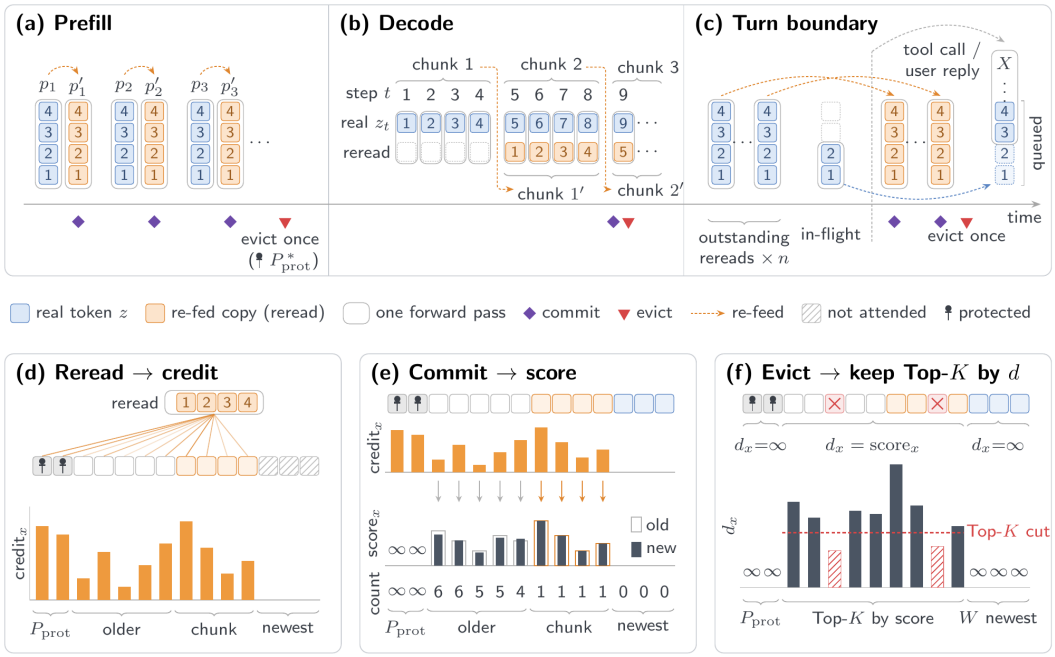


Figure 1: **Overview of MEMENTO.** *Top:* one agentic turn; each rounded outline is one forward pass, blue cells are real tokens and orange cells re-fed copies. **(a)** Each prompt chunk p_k is reread as p'_k, which sees $p_1, \ldots, p_k$. **(b)** During decoding, one re-fed token shares each real forward; every B re-feeds commit and evict. **(c)** At a turn boundary, outstanding rereads are drained, eviction runs, and the returned input X is prefilled and queued. *Bottom:* one head of the cache. **(d)** Credit accrues as a maximum over re-fed tokens; **(e)** a commit folds it into a running mean; **(f)** eviction keeps the top-K entries, exempting P_{prot} and the W newest.

Rereading is only useful if it is cheap: run as separate forwards, it doubles the decoding latency of every turn. Yet agentic context arrives as a stream the model is already decoding (Duong et al., 2026), and each decoding step is bound by memory bandwidth, leaving compute idle (Pope et al., 2023), so we *batch* rereading with decoding: each finished chunk is reread one token per step, alongside the real token and under its own attention mask, while the next chunk is decoded. Scoring thus costs $1 + \epsilon$ forwards per step.

We introduce **MEMENTO**, a training-free eviction method that keeps what the model can recall. MEMENTO pairs *recall scoring*, which rereads each completed chunk and scores every cached entry by the mean attention the rereads of its own and later chunks pay to it, with *interleaved decoding*, which hides rereading inside ordinary generation. The result is a query-agnostic compressor that keeps pace with a cache that never stops growing, retaining entries a future turn can actually reach rather than entries that merely look important today.

Empirically, MEMENTO is the strongest eviction method on every benchmark we run with Muse-Glimmer-30B and Qwen3.8-27B. On SWE-bench Verified with Muse-Glimmer-30B, keeping about 23% of the cache, MEMENTO resolves 71.5% of instances, statistically indistinguishable from the 73.5% of the full KV cache, while the strongest baseline, Expected Attention, falls to 65.0%. On BrowseComp-Plus, where the budget keeps about 12% of a Muse-Glimmer-30B session, it answers 51.0% with Muse-Glimmer-30B and 48.5% with Qwen3.8-27B, against 55.0% and 58.0% for the full KV cache and 13.5 and 10.5 points ahead of the best baseline. On τ^2-bench it leads the strongest baseline by 7 to 42 points in every domain and matches the full KV cache on Muse-Glimmer-30B retail.

Our contributions are fourfold:

- **Accessibility as the retention criterion.** Under a fixed budget, the quantity to preserve is not an entry's importance but whether attention can reach it when a future query arrives.
- **Rereading as a test of recall.** Rereading a completed chunk turns the model's own attention into a test of recall over the cache (Section 2).
- **Interleaved decoding.** Rereading shares the forwards that decode the next chunk, so scoring costs $1 + \epsilon$ forwards per step rather than 2.
- **Evidence and analysis.** On real agent sessions, query-dependent selection evicts KV that later turns need, and the loss grows over turns (eight turns later it keeps 69.6% of the KV they attend to most, against 78.0% with reread scores; Section 3.1), and a geometric analysis gives a condition under which rereading beats query-dependent eviction (Section 3.2).

Algorithm 1 MEMENTO DECODE: one iteration per decoding step.

Require: chunk size B; budget K; window W
Require: state $\mathcal{C}$, score, count, Pending, Buffer, Reread
for each decoding token z of the turn **do**
  if Reread $=$ NONE **and** Pending $\neq \emptyset$ **then**
    STARTREREAD(Pending.pop())
  batch $\leftarrow [(z, \mathcal{C})]$
  if Reread $\neq$ NONE **then**
    batch.append(Reread.NEXTTOKEN())
  $[\ell^{\text{real}}, a] \leftarrow f(\text{batch})$; $z' \sim \ell^{\text{real}}$
  append the KV of z to $\mathcal{C}$
  if Reread $\neq$ NONE **then**
    Reread.credit$_x \leftarrow \max(\text{Reread.credit}_x, a_x)$
      for all $x \in \mathcal{C}_{\leq \text{Reread.end}}$
    append the re-fed token's KV to Reread.KV
    Reread.pos $\leftarrow$ Reread.pos $+ 1$
    if Reread.pos $> B$ **then** COMMITRECALL(); EVICT()
  end if
  count$_z \leftarrow 0$; Buffer.append(z)
  if |Buffer| $= B$ **then** Pending.push(Buffer);
    Buffer $\leftarrow$ []
end for
procedure STARTREREAD(chunk)
  Reread $\leftarrow$ {chunk, pos $= 1$, end $=$ end(chunk),
    KV $= \emptyset$, credit $\equiv 0$}
procedure Reread.NEXTTOKEN()
  return (Reread.chunk[Reread.pos],
    $\mathcal{C}_{\leq \text{Reread.end}} \cup$ Reread.KV)

procedure COMMITRECALL()
  for $i \in$ Reread.chunk: score$_i \leftarrow$ Reread.credit$_i$, count$_i \leftarrow 1$
  for finite-scored older j:
    $\text{score}_j \leftarrow \frac{\text{count}_j \cdot \text{score}_j + \text{Reread.credit}_j}{\text{count}_j + 1}$
    count$_j \leftarrow$ count$_j + 1$
  Reread $\leftarrow$ NONE
procedure EVICT() `# Definition 3`
  if $|\mathcal{C}| > K$ **then** $\mathcal{C} \leftarrow$ TOPK$(\mathcal{C}, K;$ key $= d)$

2 MEMENTO: STREAMING MEMORY WITH INTERLEAVED DECODING

We introduce MEMENTO, a query-agnostic KV cache compression method for multi-turn inference. A *turn* consists of an incoming user or tool input followed by a decoding episode, and a *session* is a sequence of turns that share one cache. Text enters the KV cache $\mathcal{C}$ of the model f in blocks of B tokens, called *chunks*, whether it is generated by the model or returned by a tool. Under a fixed memory budget, MEMENTO must select which cached entries to retain without access to the turns that will later attend to them. Only a *protected prefix* P_{prot} is exempt and never evicted. Algorithm 1 gives the decoding loop; the prefill of the first prompt and the turn boundaries reuse its components (Appendix A).

Score as a reader, not a writer. An entry is worth keeping if a future turn will retrieve it, yet that turn's queries do not exist yet. Like the writer who rereads what was just written (Section 1), MEMENTO gives its tokens two roles. During decoding, a token is a writer: the token it predicts does not exist yet, so its query's attention weights mix everything that shapes the prediction, from syntax and recent tokens to attention sinks. Re-feeding a completed chunk turns the same tokens into readers. The tokens they would predict already exist in the context, so their queries are left to retrieval: the weights sharpen onto the KV entries that the chunk's content draws on. **Because a session is stationary, its next turns query the cache with content like its recent chunks, so the entries recent rereads attend to are the ones those turns will retrieve.**

The reread. Follow one chunk through a turn. As the model decodes, its tokens are written to $\mathcal{C}$ and collected in a buffer; once B of them have accumulated, the chunk is complete and joins a first-in, first-out queue of chunks waiting to be reread. When it reaches the head of the queue, MEMENTO feeds the chunk to the model a second time, placing the copies immediately after the originals, and records how strongly each cached entry is attended to by the re-fed queries.

Definition 1 (Reread)**.** Let $c = (c_1, \ldots, c_B)$ be a completed chunk whose last token sits at position $\mathrm{end}(c)$, and let $\mathcal{C}_{\leq \mathrm{end}(c)}$ be the cached entries at positions up to $\mathrm{end}(c)$. The *reread* of c feeds $c_1, \ldots, c_B$ to f again at positions $\mathrm{end}(c)+1, \ldots, \mathrm{end}(c)+B$. We write $\mathrm{Reread.KV}_{\leq r}$ for the KV of the first r re-fed copies. In each full-attention layer and KV head, the query q_r of the r-th copy gives every $x \in \mathcal{C}_{\leq \mathrm{end}(c)}$ the weight

$$a_{r,x} = \frac{\exp(q_r^\top k_x/\sqrt{d})}{\sum_{j \in \mathcal{C}_{\leq \mathrm{end}(c)} \cup \mathrm{Reread.KV}_{\leq r}} \exp(q_r^\top k_j/\sqrt{d})}, \qquad \mathrm{credit}_x = \max_{1 \leq r \leq B} a_{r,x},$$

and credit_x is the reread's *credit* for x. Under grouped-query attention the maximum also runs over the query heads that share a KV head. The re-fed KV never enters $\mathcal{C}$, and the logits of the re-fed tokens are discarded.

Many rereads, one score. MEMENTO rereads every chunk, so each cached entry receives credit from many rereads: first the reread of its own chunk, then the reread of every later chunk while it stays cached. Its score is the mean of these credits.

Definition 2 (Recall score)**.** For a cached entry x, let $\mathcal{T}_x$ be the committed rereads whose view $\mathcal{C}_{\leq \mathrm{end}(c)}$ contained x: the reread of its own chunk and every later reread while x stays cached. With $\mathrm{count}_x = |\mathcal{T}_x|$ and $\mathrm{credit}_x^{(t)}$ the credit of reread t, the *recall score* of x is

$$\mathrm{score}_x = \begin{cases} +\infty & \text{if } x \in P_{\mathrm{prot}} \text{ or } \mathrm{count}_x = 0, \\ \dfrac{1}{\mathrm{count}_x} \displaystyle\sum_{t \in \mathcal{T}_x} \mathrm{credit}_x^{(t)} & \text{otherwise.} \end{cases} \tag{1}$$

The mean is maintained incrementally: a newly generated entry starts with $\mathrm{count} = 0$ and is exempt from eviction until its first reread commits, and each committed reread folds its credit into the running mean of every entry in its view.

Letting go. Scores change only when a reread commits, so this is when MEMENTO decides what to evict. After folding in the new credits, it checks the budget and, if the cache exceeds it, evicts the lowest-scored entries. The W newest entries are exempt: no reread has reached them yet, and the writer is the most likely to attend to them in the next steps.

Definition 3 (Eviction)**.** Let the *budget* K be the number of entries the cache may hold across all full-attention layers and KV heads, reported per KV head as M, and let W be the recency window. Whenever $|\mathcal{C}| > K$, MEMENTO ranks entries by the decision score

$$d_x = \begin{cases} +\infty & \text{if } x \text{ is among the } W \text{ newest entries of its head,} \\ \mathrm{score}_x & \text{otherwise,} \end{cases}$$

and keeps the highest-ranked entries the budget allows, $\mathcal{C} \leftarrow \textsc{TopK}\big(\mathcal{C},\ K;\ \mathrm{key} = d\big)$.

Reading while writing. Run as separate forwards, the rereads would double the decoding latency of a turn. A decoding step, however, processes a single token and is bound by memory bandwidth,

leaving room for one more token at little cost, much as writers reread on capacity that typing leaves idle (Kellogg, 1996; Olive, 2014; Alamargot et al., 2007). MEMENTO therefore batches the reader with the writer: at every step the real token z and the next re-fed copy c_r share a **single batched forward** (Algorithm 1, lines 4–7), each with its own position and mask. Only z attends to all of $\mathcal{C}$, emits logits and enters the cache; c_r attends to $\mathcal{C}_{\leq \mathrm{end}(c)}$ and the copies before it, and only updates the credits. A linear-attention layer, which has no entries to mask, gives the reader its own copy of its state at $\mathrm{end}(c)$. A chunk is thus reread over the B steps in which the next chunk is written. Because the reader reuses the weight and cache reads the writer already performs, **scoring costs** $1+\epsilon$ **forwards per step rather than** 2; the savings come from batching, not from less arithmetic, and depend on hardware and workload.

Reading without a writer. At two points no decoding token is available to batch the reader with. At the start of a session, the first prompt P is split into chunks of B_0 tokens and reread chunk by chunk, the copy of each chunk attending only to that chunk and the ones before it, at a one-time cost of $|P|$ auxiliary tokens. At a turn boundary, the outstanding rereads are completed with dedicated forwards, eviction runs once, and the returned input is prefilled and queued for rereading during the next turn; chunks not reread within the turn stay unscored, and so exempt from eviction, until the next boundary (Algorithms 2 and 3, Appendix A).

3 Why Rereading Works

3.1 Empirical Evidence: Rereading Predicts Future Recall

Setup. We record full-cache agent sessions of Muse-Glimmer-30B on BrowseComp-Plus and τ^2-bench retail. A *turn* reads one input (a user message or tool result) and writes a reply. After each turn, every method keeps $M = 4096$ KV per head out of the evictable cache I_t, using only what has been generated so far. We then check how well the kept KV serve the next turn, which the full cache lets us observe. As references we add two methods that see the future. *Query-dependent* scores KV by the attention of the next turn's input only (the user message or tool result); it sees the request but not the reply, so it is a half-oracle rather than a baseline (marked †). *Oracle* scores KV by the attention of the whole next turn, including the reply it writes, an upper bound.

F1: Key and value statistics do not tell which KV the next turn will read; rereading does. Keeping KV by key norm, value norm or key distinctiveness matches the next turn's top KV no better than random (21.5–30.8% vs 29.0%, Table 1a). MEMENTO reaches 56.1% without seeing the request, close to the half-oracle query-dependent score (61.1%).

F2: Even with the request in hand, query-dependent eviction drops KV that later turns need. Scoring by the request keeps what it needs, but evicted KV never come back. Table 1b takes the KV that each decoding query (per token and head) attends to most k turns after an eviction, and counts how often it is still kept. Eight turns later, query-dependent eviction keeps 69.6% of them and MEMENTO keeps 78.0%. The missing share is the π of Theorem 1.

Table 1: **What each eviction rule keeps.** Muse-Glimmer-30B, BrowseComp-Plus, $M = 4096$.

(a) Overlap (%) of the kept KV with the KV kept by Oracle (the M KV the next turn attends to most).

Eviction rule	Overlap
Random	29.0
Key norm	29.4
Value norm	30.8
Key distinctiveness	21.5
MEMENTO	**56.1**
Query-dependent†	61.1
Oracle	100

(b) Kept rate (%): share of the most-attended KV k turns after an eviction that are still kept. Higher is better.

	k=1	2	4	8
Query-dependent†	88.4	85.5	81.1	69.6
MEMENTO	**89.2**	**87.2**	**84.3**	**78.0**

F3: Without seeing the request, MEMENTO keeps nearly as much next-turn attention as the half-oracle. On BrowseComp-Plus, MEMENTO keeps 65.1% of the next turn's attention, about twice StreamingLLM, KeyDiff and LagKV (19.8–33.2%) and close to query-dependent (68.0%), which sees the request (Table 2). On τ^2-bench MEMENTO is within one point of the half-oracle baseline (88.4% vs 89.2%).

Table 2: **Next-turn attention kept.** Share (%) of the next turn's attention on the evictable cache that falls on the kept KV by each eviction rule, $M = 4096$. Higher is better. †Sees the next request.

Eviction Rule	Oracle	Random	Streaming LLM	KeyDiff	LagKV	Query-dependent†	MEMENTO
BrowseComp-Plus	82.6	29.1	33.2	19.8	28.2	68.0	65.1
τ^2-bench	96.4	58.2	67.6	46.6	65.0	89.2	88.4

3.2 Theoretical Analysis: Reread Witnesses of Retrieval Cells

We interpret rereading through the geometry of maximum-inner-product retrieval, for one layer and one head with fixed query and KV representations; formal statements and proofs are in Appendix B. The keys of a cache partition query space into *retrieval cells*: the cell $\mathcal{C}_i$ of entry i is the set of queries whose largest logit is on i, a power cell with site k_i and weight $\|k_i\|_2^2$ (Aurenhammer, 1987). A reread query that lands in $\mathcal{C}_i$ and gives i attention at least the eviction threshold τ *witnesses* that cell, and one strong witness suffices to retain i. A future query need not resemble its witness; it only needs to fall in a witnessed cell with a logit margin γ over every other entry. We call the union of these regions the covered region $\Omega_{\tau,\gamma}$.

Results. For a query in $\Omega_{\tau,\gamma}$, evicting m entries leaves the winning key unchanged and moves the attention output by at most $2V\, me^{-\gamma}/(1 + me^{-\gamma})$, where $\|v_i\|_2 \le V$; over a future-query distribution, the expected error adds at most $2V$ times the uncovered probability $\delta_{\tau,\gamma}$ (Proposition 1). Theorem 1 compares this bound with a query-dependent policy that keeps the same number of entries, chosen by the current request. Suppose that with probability π a later query puts attention at least $1 - \varepsilon_D$ on an entry that the policy discarded, whose value lies at distance at least Δ from the convex hull of the retained values. Then the policy's expected error is at least $\pi(\Delta - 2V\varepsilon_D)_+$, and rereading has the lower expected future-query error whenever its bound falls below this one. The comparison is conditional: when the current request predicts future needs well, π is small and query-dependent selection can win. Because eviction is irreversible, reselecting at the next turn cannot recover what an earlier turn evicted, so the per-turn advantage accumulates over a session (Appendix B.6).

4 Experiments

We evaluate MEMENTO on three models and three agentic benchmarks, BrowseComp-Plus (web search), τ^2-bench (customer service) and SWE-bench Verified (software engineering), in the streaming regime: every method evicts during prefill and decoding, except the protected prefix, and an evicted entry never returns. Appendix C gives the full protocol and additional results.

4.1 Setup

Models. Muse-Glimmer-30B (Meta Superintelligence Lab, 2026), Qwen3.8-27B (Qwen Team, 2026) and Qwen3-8B (Yang et al., 2025), with reasoning on and seed 42; only full-attention layers, whose cache grows with the session, are scored and evicted (Appendix C.1).

Benchmarks and metrics. BrowseComp-Plus (Chen et al., 2025), τ^2-bench (Barres et al., 2025) and SWE-bench Verified (Jimenez et al., 2024), the last run on Muse-Glimmer-30B and Qwen3-8B. We report accuracy judged by Qwen3-32B on BrowseComp-Plus, pass^1 on τ^2-bench and the resolved rate on SWE-bench Verified; the appendix adds KV/head, the mean number of entries per KV head on the compressed layers. Paired differences use an exact two-sided McNemar test.

Baselines. The reference is the full KV cache. The four baselines are training-free, query-agnostic scorers, each ranking an entry by what it contains or where it sits: StreamingLLM (Xiao et al., 2024b) by position, KeyDiff (Park et al., 2025) by key geometry, Expected Attention (Devoto et al., 2025) by attention statistics and LagKV (Liang et al., 2025) by lag-relative key and value statistics. **None of them runs retrieval.** All four evict in the same streaming regime, at the same budget and with the same protected prefix. Query-dependent methods, which Section 3 argues against, are not compared end to end. FastKVzip (Kim et al., 2026), which trains a gate to predict the reconstruction attention of KVzip (Kim et al., 2025), is not training-free; we compare it with MEMENTO in Appendix C.5.

Table 3: **Main results.** BrowseComp-Plus accuracy (%; 95% half-widths at most 6.9 pp) and τ^2-bench pass[1] (%, one trial per task; 114 / 114 / 97 tasks); $M = 4096$, $B = 128$, $W = 256$. Muse: Muse-Glimmer-30B; Qwen: Qwen3.8-27B. Bold: best eviction method per column. Appendix Tables 7 and 8 add Qwen3-8B, counts and cache size.

	BrowseComp-Plus		τ^2-bench, Muse			τ^2-bench, Qwen		
Method	Muse	Qwen	retail	telecom	banking	retail	telecom	banking
full KV cache	55.0	58.0	83.3	92.1	43.3	86.0	93.0	42.3
StreamingLLM	5.0	19.5	36.8	23.7	5.2	7.0	14.9	7.2
KeyDiff	11.5	8.0	49.1	50.0	11.3	20.2	33.3	11.3
Expected Attention	37.5	38.0	21.9	22.8	9.3	6.1	34.2	13.4
LagKV	9.0	28.5	47.4	33.3	9.3	21.9	17.5	12.4
MEMENTO	**51.0**	**48.5**	**83.3**	**75.4**	**18.6**	**64.0**	**76.3**	**20.6**

Table 4: SWE-bench Verified on Muse-Glimmer-30B, 200 instances; $M = 8192$. Resolved: % (count). Sessions ending by a submission (the only source of a patch), at the context limit or at the 250-step limit; Steps: median LLM calls. Appendix Table 9 adds Qwen3-8B.

Method	Resolved	Submitted	Ctx. limit	Step limit	Steps
full KV cache	73.5 (147)	199	0	1	74
Expected Attention	65.0 (130)	164	10	26	87
MEMENTO	**71.5 (143)**	**189**	**1**	**10**	**74**

MEMENTO. We use $M = 4096$ (8192 on SWE-bench Verified), $B = 128$, $W = 256$ and $B_0 = 2048$; the real and the re-fed token of a decoding step share one forward pass, and COMMITRECALL keeps the mean of all credits.

4.2 MAIN RESULTS

BrowseComp-Plus. The agent must combine evidence from documents retrieved many searches apart, so a document's value is unknown when it enters the cache, and the answer format sits in a short first prompt that must stay accessible to the end. A Muse-Glimmer-30B session grows to 35k tokens on average, so the budget keeps about 12% of the cache. MEMENTO nearly matches the full KV cache with 12% of its entries (Table 3): it answers 102/200 queries against 110/200, keeping 93% of the correct answers, and is the only eviction method within 5 points. The strongest baseline, Expected Attention, answers 75/200, and MEMENTO wins 35 of the queries that only one of the two solves against 8 ($p < 0.001$). 93.5% of MEMENTO's answers follow the required format, against 49.0% for Expected Attention (Appendix C.2). The lead holds on Qwen3.8-27B, where MEMENTO answers 97/200, 21 queries ahead of Expected Attention (33 to 12, $p = 0.002$), though 19 short of the full KV cache ($p = 0.001$).

τ^2-bench. The rules the agent must follow sit in a long first prompt, while the requests that decide which rules matter arrive only in later turns, and one evicted rule can fail a task. The policy, tool schemas and, on banking, the task's reference documents exceed M in the first prompt of all but 4 of the 325 tasks, so eviction starts at prefill, before any request is known. MEMENTO is the best eviction method in every domain on both models, 7 to 42 points ahead of the strongest baseline. On Muse-Glimmer-30B it matches the full KV cache on retail (95 of 114 tasks each), while no baseline passes 49.1 / 50.0 / 11.3% (retail / telecom / banking); on Qwen3.8-27B it is about three times the best baseline on retail and more than twice on telecom. The remaining gap to the full KV cache, largest on telecom and banking, comes from the first prompt: its entries are scored once at prefill, before any request arrives, and what that eviction drops no later reread can restore.

SWE-bench Verified. Here the agent's own output is the largest part of the context, and the file the fix touches is first read at a median step 3, so what the agent reads early is needed for most of the session. A Muse-Glimmer-30B session runs a median of 74 LLM calls and ends at 35k tokens, so the budget keeps about 23% of the cache. MEMENTO nearly matches the full KV cache (Table 4): it resolves 143/200 instances against 147/200 (3 wins to 7, $p = 0.34$), keeping 97% of the resolved

instances with 23% of the entries; 189 of 200 sessions submit, against 199, and the median session is 74 calls long in both. Expected Attention, the strongest baseline, resolves 130/200 and loses to MEMENTO 5 to 18 ($p = 0.011$): 36 of its sessions end at the step or context limit without a patch, against 11 for MEMENTO.

Speed. On our vLLM implementation with Qwen3-8B, a fixed 24 GiB HBM budget holds one 28.7k-token request under full attention and six under MEMENTO, for $3.17\times$ the decoding throughput; the gain needs long sequences (Appendix C.6).

4.3 Ablations

Score rule. The ablations use Muse-Glimmer-30B at the default setting unless stated. CommitRecall decides how the credits of successive rereads merge into one score; Table 5 compares the mean of Eq. (1) with four alternatives on BrowseComp-Plus, among them *no-refresh* (own-reread credit only). A KV's own-reread credit is about 140 times a later reread's (medians 0.19 and 0.0013; 78 times on Qwen3-8B), so how a rule weighs the two decides whether later cues can move a KV at all. **The mean puts every credit on one scale and keeps the score a per-reread recall strength, so a KV is judged by how well cues recall it.** It is the best rule by a wide margin (102/200). The maximum is stuck on the own credit, which a later credit almost never exceeds; the moving average forgets old credits; and own credit plus the mean of later ones, like no-refresh, leaves the own scale dominant. On τ^2-bench the mean and no-refresh are within a few tasks of each other (Table 3; $p \geq 0.30$), as the first-prompt argument of Section 4.2 explains.

Which heads may score. Each KV head keeps the positions its own credits select. Imposing one shared set of positions instead, chosen by the 136 retrieval heads of a copy test (Wu et al., 2025) or by the complementary 280 heads, loses 13 to 23 BrowseComp-Plus queries under the mean rule ($p \leq 0.035$) and 17 to 67 τ^2-bench retail tasks ($p \leq 0.004$), while which heads choose matters little (Appendix C.7, Table 11): the query that reaches an entry is head-specific.

Table 5: **Score rule.** Muse-Glimmer-30B, BrowseComp-Plus (correct of 200). Rules differ only in how CommitRecall merges the credits of successive rereads.

Rule	Correct
Mean (MEMENTO)	**102**
Maximum	37
EMA (γ=0.9)	67
Own + mean of later	20
No-refresh (own only)	24

Prefill reread vs. KVzip-style scoring. MEMENTO rereads the first prompt chunk by chunk (Algorithm 2); each reread scores every KV of the context read so far, and a KV's score averages all the rereads that see it. KVzip (Kim et al., 2025) instead scores each KV once, only by the reconstruction of its own copy. Swapping in KVzip's scoring for the prompt, with everything else unchanged, costs Muse-Glimmer-30B 16 telecom tasks (70 vs 86, $p = 0.007$) and leaves banking unchanged (18 vs 18); on Qwen3.8-27B the two are within noise ($p \geq 0.74$; Table 12). MEMENTO's scoring is never worse, is clearly better once, and lets one rule score both the prompt and the stream.

Budget, window and block size. Figure 2 varies one knob at a time around the default, for MEMENTO and for no-refresh. In MEMENTO every later reread refreshes the score of the older KV it sees (the mean of all credits); no-refresh scores each KV once, by its own reread, and never updates it. The budget is the only strongly sensitive knob, and refresh is what makes it a mild one. Without refresh, accuracy rises from 13.0% at M=4096 to 55.5% at M=16384 (88 wins to 3, $p < 0.001$) and then saturates (56.0% at M=24576); with refresh the same budget only takes it from 47.0% to 55.0% ($p = 0.004$), and the two rules meet at M=16384. At M=2048 both collapse (28.0% and 5.0%, lower bounds). The recency window and the block size move the full method nowhere ($p \geq 0.47$ and $p \geq 0.15$), whereas without refresh a larger window hurts (W=1024 and 2048: $p = 0.036$ and 0.006), since the rows it shields are rows the budget cannot spend on evidence. The block size changes no accuracy for either rule ($p \geq 0.08$) but sets the cost of scoring: 8.2 minutes per query at B=32 against 6.3 at B=512.

5 Related Work

KV cache eviction. Most training-free methods score entries by the attention of the current query or a recent window. H2O keeps accumulated heavy hitters (Zhang et al., 2023), TOVA treats the

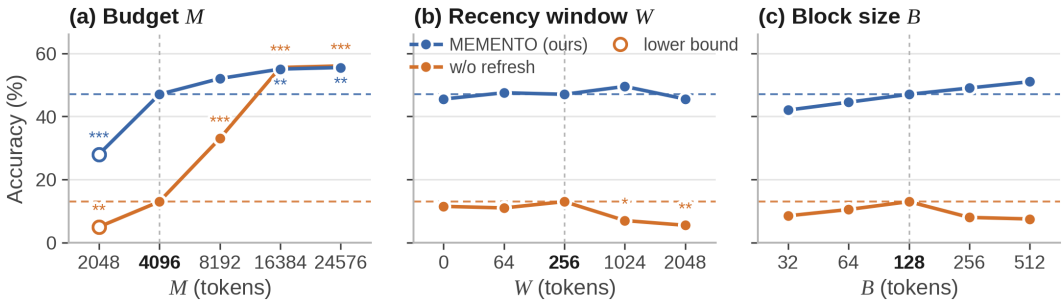


Figure 2: **Budget, window and block size.** Muse-Glimmer-30B on BrowseComp-Plus (200 queries), one knob varied at a time around the default M=4096, W=256, B=128 (bold tick), for MEMENTO and for the no-refresh rule; the horizontal dashed lines are each rule at that default, re-run inside the sweep (94/200 and 26/200). Stars mark arms that differ from that re-run under an exact McNemar test ($^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$); hollow markers are lower bounds (Appendix C.7).

bounded cache as a recurrent memory (Oren et al., 2024), SnapKV scores the prompt with a trailing observation window (Li et al., 2024), and PyramidKV and Ada-KV reallocate the budget across layers and heads (Cai et al., 2024; Feng et al., 2025). When compression repeats across turns, these scores specialize the cache to the requests already seen (Section 3.2). Query-agnostic scores avoid this: StreamingLLM keeps attention sinks and a recent window (Xiao et al., 2024b), KeyDiff scores keys by their geometric distinctiveness (Park et al., 2025), LagKV by lag-relative statistics (Liang et al., 2025), and Expected Attention estimates the attention of future queries in closed form (Devoto et al., 2025). None of them tests whether attention reaches an entry.

KVzip. Closest to MEMENTO, KVzip scores entries by the attention of a context reconstruction pass (Kim et al., 2025). It scores once, at prefill, so it does not cover context that arrives during decoding, and each entry only by the reconstruction of its own chunk; MEMENTO rereads every new chunk and averages the credit of every reread that sees an entry. FastKVzip (Kim et al., 2026) trains a gate to predict KVzip's scores and can evict during decoding, yet it falls far behind the training-free MEMENTO on BrowseComp-Plus and SWE-bench Verified (Appendix C.5).

Other directions. Quest and MInference select entries per query from a full cache (Tang et al., 2024; Jiang et al., 2024), KIVI quantizes entries (Liu et al., 2024b), and serving systems and agent memories manage context in blocks or text segments (Kwon et al., 2023; Zheng et al., 2024; Packer et al., 2023); see Appendix D.

6 Conclusion

We argued that a bounded KV cache for multi-turn inference should keep what the model can recall, not merely what it has stored, and introduced MEMENTO, a training-free eviction method that rereads each completed chunk to score cached entries by whether attention actually reaches them, at $1+\epsilon$ forwards per decoding step. On every benchmark we run with Muse-Glimmer-30B and Qwen3.8-27B, MEMENTO is the strongest eviction method and nearly matches the full KV cache on BrowseComp-Plus and SWE-bench Verified with 12% and 23% of its entries. **Limitations.** Every arm is a single agent run (95% half-width 3 to 7 pp on BrowseComp-Plus and SWE-bench Verified), so gaps of a few queries are unresolved and we rely on paired tests; SWE-bench Verified, at 25 GPU-hours per 200-instance arm, is run on 200 of its 500 instances, at one budget and with one of the four baselines. Extending rereading to the first prompt before any request arrives, and integrating it into production serving engines, are natural next steps.

AI use statement

Generative AI tools were used only to polish the writing of this paper; the authors are responsible for all content.

Ethics statement

This work studies training-free KV cache compression for language models and does not involve human subjects, personal data, or new datasets. All benchmarks used are publicly released.

Reproducibility statement

Section 2, Algorithm 1 and Appendix A specify the full method. Appendix C.1 lists models, harnesses, sampling settings and all hyperparameters (seed 42 throughout), and Appendix B gives all proofs. We are unable to release code at submission time because it is pending our organization's legal approval; we will release the code upon approval should the paper be made public.

References

Denis Alamargot, Christophe Dansac, David Chesnet, and Michel Fayol. Parallel processing before and after pauses: A combined analysis of graphomotor and eye movements during procedural text production. In Mark Torrance, Luuk van Waes, and David Galbraith (eds.), *Writing and Cognition: Research and Applications*, pp. 11–29. Elsevier, 2007. doi: 10.1163/9781849508223_003.

Anthropic. Claude code. `https://claude.com/product/claude-code`, 2025.

Franz Aurenhammer. Power diagrams: Properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987. doi: 10.1137/0216006.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov. RocketKV: Accelerating long-context LLM inference via two-stage KV cache compression. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL `https://arxiv.org/abs/2502.14051`.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024. URL `https://arxiv.org/abs/2406.02069`.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

Vin de Silva and Gunnar Carlsson. Topological estimation using witness complexes. In *Symposium on Point-Based Graphics*, pp. 157–166. The Eurographics Association, 2004. doi: 10.2312/SPBG/SPBG04/157-166.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: KV cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025. URL `https://arxiv.org/abs/2510.00636`.

Duc Duong, Hoang Anh Duy Le, Jianwen Xie, Anshumali Shrivastava, and Zhaozhuo Xu. Forget without compromise: Nexus sampling for streaming KV-cache eviction under fixed budgets. *arXiv preprint arXiv:2606.23961*, 2026. URL `https://arxiv.org/abs/2606.23961`.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems*, 2025. URL `https://arxiv.org/abs/2407.11550`.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations*, 2024a. URL `https://arxiv.org/abs/2310.01801`.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations*, 2024b. URL https://arxiv.org/abs/2307.06945.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In *Proceedings of Machine Learning and Systems*, 2024. URL https://arxiv.org/abs/2311.04934.

Google. Google antigravity. https://antigravity.google/, 2025.

John R. Hayes and Linda S. Flower. Identifying the organization of writing processes. In Lee W. Gregg and Erwin R. Steinberg (eds.), *Cognitive Processes in Writing*, pp. 3–30. Lawrence Erlbaum, Hillsdale, NJ, 1980.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, 2023. doi: 10.18653/v1/2023.emnlp-main.825.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2407.02490.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.06770.

Ronald T. Kellogg. A model of working memory in writing. In C. Michael Levy and Sarah Ransdell (eds.), *The Science of Writing: Theories, Methods, Individual Differences, and Applications*, pp. 57–71. Lawrence Erlbaum, Mahwah, NJ, 1996.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.23416.

Jang-Hyun Kim, Dongyoon Han, and Sangdoo Yun. Fast KVzip: Efficient and accurate LLM inference with gated KV eviction. *arXiv preprint arXiv:2601.17668*, 2026. URL https://arxiv.org/abs/2601.17668.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Hoang Anh Duy Le, Sahil Joshi, Zeyu Yang, Zhaozhuo Xu, and Anshumali Shrivastava. Scout before you attend: Sketch-and-walk sparse attention for efficient LLM inference. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.07397.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2404.14469.

Manlai Liang, JiaMing Zhang, Xiong Li, and Jinlong Li. LagKV: Lag-relative information of the KV cache tells which tokens are important. *arXiv preprint arXiv:2504.04704*, 2025. URL https://arxiv.org/abs/2504.04704.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a. doi: 10.1162/tacl_a_00638.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.17118.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *International Conference on Machine Learning*, 2024b. URL https://arxiv.org/abs/2402.02750.

Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/TPAMI.2018.2889473.

Meta Superintelligence Lab. Introducing Muse Glimmer: An open agentic model that runs on your device. https://research.meta.ai/blog/introducing-muse-glimmer-open-agentic-model, August 2026. Model card at https://huggingface.co/meta-models/Muse-Glimmer-30B.

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.16300.

Moonshot AI. Kimi K3. https://www.kimi.ai/blog/kimi-k3, 2026.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2304.08467.

Thierry Olive. Toward a parallel and cascading model of the writing system: A review of research on writing processes coordination. *Journal of Writing Research*, 6(2):173–194, 2014. doi: 10.17239/jowr-2014.06.02.4.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. URL https://arxiv.org/abs/2209.11895.

OpenAI. Codex. https://openai.com/codex/, 2025.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, 2024. doi: 10.18653/v1/2024.emnlp-main.1043.

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

Junyoung Park, Dalton Jones, Matthew J. Morse, Raghavv Goel, Mingu Lee, and Chris Lott. KeyDiff: Key similarity-based KV cache eviction for long-context LLM inference in resource-constrained environments. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2504.15364.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 5, pp. 606–624, 2023.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork. https://qwen.ai/blog?id=qwen3.8, August 2026. Release blog of the Qwen3.8 family; Qwen3.8-27B weights at https://huggingface.co/Qwen/Qwen3.8-27B.

Jeroen G. W. Raaijmakers and Richard M. Shiffrin. Search of associative memory. *Psychological Review*, 88(2):93–134, 1981. doi: 10.1037/0033-295X.88.2.93.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2303.11366.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, and Gongyi Wang. RazorAttention: Efficient KV cache compression through retrieval heads. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.15891.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2406.10774.

Endel Tulving and Zena Pearlstone. Availability versus accessibility of information in memory for words. *Journal of Verbal Learning and Verbal Behavior*, 5(4):381–391, 1966. doi: 10.1016/S0022-5371(66)80048-8.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2404.15574.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems*, 2024a. URL https://arxiv.org/abs/2402.04617.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024b. URL https://arxiv.org/abs/2309.17453.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10819.

Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, Jian-Guang Lou, and Shuai Ma. Re-reading improves reasoning in large language models. *arXiv preprint arXiv:2309.06275*, 2023. URL https://arxiv.org/abs/2309.06275.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 94–109, 2025. doi: 10.1145/3689031.3696098.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024. URL https://arxiv.org/abs/2406.12045.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.14048.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2312.07104.

A Prefill and Turn Boundaries

Algorithm 1 covers the decoding steps of a turn. A session runs Algorithm 2 once on the first prompt P, then alternates Algorithm 1 with Algorithm 3 at each turn boundary, where the returned input X (a tool result or the forced-answer directive) enters the cache. Both reuse StartReread, Reread.nextToken, CommitRecall and Evict from Algorithm 1. We write $f(x; \mathcal{K})$ for a forward of f on the tokens x that attends to the cache $\mathcal{K}$ and the causal prefix of x and appends the KV of x to $\mathcal{K}$; $f(P)$ starts from an empty cache.

Prefill. A causal forward builds the prompt KV, and the prompt is then reread in chunks of B_0 tokens, each chunk's copies seeing only the chunks before it. This is the only reread no decoding step pays for, and it costs $|P|$ auxiliary tokens once per session. Where the harness protects the whole first prompt ($P_{\text{prot}} = P$), it is never evicted and these scores decide nothing; where only the attention sinks are protected, the prompt competes on them like any other text.

Turn boundaries. Before a new input enters, the reread in flight and every queued chunk are reread with forwards of their own, since no real token is left to carry them, and eviction runs once after all of them commit. The input is then prefilled without a re-fed token or an eviction, and its complete B-token chunks are queued. Buffer stays protected and may span turn boundaries until it fills.

Backlog. During decoding one re-fed token is processed per real token and chunks are reread in FIFO order, so a chunk is scored only after the chunks queued before it. Decoding also completes a new chunk every B steps, so it never shrinks the backlog: an input of k chunks needs kB decoding steps to be scored, and the chunks the turn does not reach are scored at the next turn boundary. Until then they stay unscored and protected. The cache therefore exceeds the budget from the input until the next CommitRecall, and that eviction takes the whole excess from entries that are already scored; if the protected entries alone reach the budget, eviction is deferred until the next turn boundary.

Algorithm 2 MEMENTO Prefill: run once on the first prompt.

Require: model f; first prompt P; prefill chunk B_0; protected prefix $P_{\text{prot}} \subseteq P$
Ensure: live cache $\mathcal{C}$; score; count; empty stream state

$\mathcal{C} \leftarrow f(P)$; $\text{count}_i \leftarrow 0$ for $i \in P$
for each B_0-token chunk of P, in order **do**
  StartReread(chunk)
  batch $\leftarrow \left[\text{Reread.nextToken}() \text{ for } \text{Reread.pos} = 1, \ldots, B_0\right]$
  $\left[a_1, \ldots, a_{B_0}\right] \leftarrow f(\text{batch})$
  $\text{Reread.credit}_x \leftarrow \max_r a_{r,x}$ for all $x \in \mathcal{C}_{\leq \text{Reread.end}}$
  CommitRecall()
end for
$\text{score}_i \leftarrow +\infty$ for all $i \in P_{\text{prot}}$; Evict()
Pending $\leftarrow \emptyset$; Buffer $\leftarrow [\,]$; Reread $\leftarrow$ NONE

B Full Theoretical Analysis

This appendix gives the complete analysis summarized in Section 3.2: the formal setup and statements (Appendix B.1), the geometry of retrieval cells (Appendix B.2), the witness lemma (Appendix B.3), the proof of Proposition 1 and its corollary (Appendix B.4), the proof of Theorem 1 (Appendix B.5),

Algorithm 3 MEMENTO TURNBOUNDARY: run once when a turn's returned input X arrives.

Require: returned input X (tool result or forced-answer directive); chunk size B; state ($\mathcal{C}$, score, count, Pending, Buffer, Reread)

```
# no real tokens left to ride on, so each reread takes a forward of its own
while Reread ≠ NONE or Pending ≠ ∅ do
    if Reread = NONE then STARTREREAD(Pending.pop())
    batch ← [ Reread.NEXTTOKEN() for the re-feed positions left in Reread.chunk ]
    [a_r]_r ← f(batch)
    Reread.credit_x ← max(Reread.credit_x, max_r a_{r,x}) for all x ∈ C_{≤Reread.end}
    COMMITRECALL()
end while
if any reread committed above then EVICT()
# return prefill:  no re-fed token, no eviction
C ← f(X; C); count_i ← 0 for i ∈ X
Buffer.append(X); move every full B-token prefix of Buffer into Pending
```

and the multi-turn argument (Appendix B.6). Throughout, we analyze one layer and one head, with fixed query and KV representations at each reference cache, and use the notation of Eqs. (2)–(6).

B.1 Setup and Main Results

Retrieval cells. Let $I = \{1, \dots, N\}$ index the original entries of a reference cache, with

$$\ell_i(q) = \frac{q^\top k_i}{\sqrt{d}}, \qquad \alpha_i(q) = \frac{e^{\ell_i(q)}}{\sum_{j \in I} e^{\ell_j(q)}}, \qquad o(q) = \sum_{i \in I} \alpha_i(q) v_i. \tag{2}$$

The retrieval cell of entry i is the set of queries it wins under Eq. (2),

$$\mathcal{C}_i = \{q : \ell_i(q) \geq \ell_j(q) \text{ for all } j \in I\}. \tag{3}$$

The cells in Eq. (3) are power cells with sites k_i and weights $\|k_i\|_2^2$ (Aurenhammer, 1987), whose facet adjacency forms a weighted Delaunay graph (Appendix B.2). MEMENTO never builds this graph; its scores implicitly retain a subset of its cells.

Rereading as witnessing. For a scored entry i, let $\mathcal{R}_i$ contain the reread queries generated from its own chunk, and let $\mathcal{J}_r$ be the visible support at reread step r (the frozen original cache plus the causal auxiliary prefix). The implemented score is

$$p_r(i) = \frac{\exp(\ell_i(q_r^{\text{rep}}))}{\sum_{j \in \mathcal{J}_r} \exp(\ell_j(q_r^{\text{rep}}))}, \qquad s_i = \max_{r \in \mathcal{R}_i} p_r(i). \tag{4}$$

With eviction threshold τ and protected set P, define the retained set and the set of strongly witnessed cells,

$$\begin{aligned} S_\tau &= P \cup \{i \in I \setminus P : s_i \geq \tau\}, \\ W_\tau &= P \cup \{i \in I \setminus P : \exists r \in \mathcal{R}_i,\ q_r^{\text{rep}} \in \mathcal{C}_i,\ p_r(i) \geq \tau\}. \end{aligned} \tag{5}$$

In Eq. (5), $W_\tau \subseteq S_\tau$: because of the maximum in s_i (Eq. (4)), a single strong witness suffices for retention, and a logit margin at the reread query certifies one (Lemma 1). For $\gamma > 0$, the region of future queries covered with margin is

$$\Omega_{\tau,\gamma} = \bigcup_{i \in W_\tau} \mathcal{C}_i^\gamma, \qquad \mathcal{C}_i^\gamma = \{q : \ell_i(q) - \ell_j(q) \geq \gamma \text{ for all } j \neq i\}. \tag{6}$$

A future query in the region of Eq. (6) need not resemble its reread witness; it only needs to land in the same retained cell with margin.

Proposition 1 (Attention stability on reread-witnessed cells)**.** *Let $S = S_\tau \neq \emptyset$, $m = |I \setminus S|$, $\|v_i\|_2 \leq V$ for all i, and let $o_S(q)$ be the attention output restricted to S. For every $q \in \Omega_{\tau,\gamma}$, the winning key is unchanged by pruning and*

$$\|o(q) - o_S(q)\|_2 \leq 2V \frac{m e^{-\gamma}}{1 + m e^{-\gamma}}. \tag{7}$$

For any future-query distribution $\mathcal{D}$ with $\delta_{\tau,\gamma} = \Pr_{q\sim\mathcal{D}}[q \notin \Omega_{\tau,\gamma}]$,

$$\mathbb{E}_{q\sim\mathcal{D}}\|o(q) - o_S(q)\|_2 \le 2V\left[\delta_{\tau,\gamma} + (1-\delta_{\tau,\gamma})\frac{me^{-\gamma}}{1+me^{-\gamma}}\right]. \tag{8}$$

Eq. (8) separates missing coverage, $\delta_{\tau,\gamma}$, from the residual softmax tail inside covered cells, which Eq. (7) bounds without using any future query to select S (Corollary 1; proof in Appendix B.4).

Main theorem. Condition on a reference history H_t that includes the request at turn t, and let $\mathcal{D}_t$ describe a later query over the fixed reference cache. The reuse objective is

$$\mathcal{L}_t(S) = \mathbb{E}_{q\sim\mathcal{D}_t}\big[\|o(q) - o_S(q)\|_2\big].$$

A query-dependent policy selects S_D from current-turn cues, while MEMENTO selects $S_R = S_\tau$; both keep the same number of entries.

Theorem 1 (A sufficient condition for lower future-query error)**.** *Let $|S_D| = |S_R| \ge 1$ and $\|v_i\|_2 \le V$. Let δ_R be the uncovered probability of $\Omega_{\tau,\gamma}$ under $\mathcal{D}_t$, and define*

$$e_R = \frac{(|I| - |S_R|)e^{-\gamma}}{1 + (|I| - |S_R|)e^{-\gamma}}, \qquad U_R = 2V[\delta_R + (1-\delta_R)e_R].$$

Suppose that with probability at least π, a future query has some $i = i(q) \notin S_D$ with

$$\alpha_i(q) \ge 1 - \varepsilon_D, \qquad \operatorname{dist}(v_i, \operatorname{conv}\{v_j : j \in S_D\}) \ge \Delta, \tag{9}$$

where $0 \le \varepsilon_D < 1$ and $\Delta > 0$. With $L_D = \pi(\Delta - 2V\varepsilon_D)_+$, we have $\mathcal{L}_t(S_R) \le U_R$, $\mathcal{L}_t(S_D) \ge L_D$, and hence

$$\boxed{U_R < L_D \quad\Longrightarrow\quad \mathcal{L}_t(S_R) < \mathcal{L}_t(S_D).} \tag{10}$$

Here π measures future demand for entries that current-query selection discards, and Δ in Eq. (9) prevents that demand from being met by redundant retained values. The reread bound depends on the probability mass of the covered cells, not their count. The comparison in Eq. (10) is conditional (proof and discussion in Appendix B.5): when the current query accurately predicts future needs, query-dependent selection can win. The claim is robustness to a shift in retrieval needs, not dominance for every future-query distribution.

Eviction is irreversible. Let I_t be the entry identities introduced by turn t and $C_t \subseteq I_t$ the retained ones. If turn $t+1$ adds only new identities A_{t+1} and discarded entries are never recomputed, then

$$I_{t+1} = I_t \cup A_{t+1}, \qquad C_{t+1} \subseteq C_t \cup A_{t+1}, \qquad I_t \setminus C_t \subseteq I_{t+1} \setminus C_{t+1}. \tag{11}$$

By Eq. (11), re-running query-dependent selection at the next turn cannot recover what an earlier turn evicted, so the per-turn advantage of Theorem 1 accumulates over a session (Eq. (12), Appendix B.6). Rereading aims to make the selection before this irreversible step better suited to future reuse.

B.2 Attention Cells and Their Delaunay Dual

Queries and keys include their positional transformations. Inside the retrieval cell $\mathcal{C}_i$ of Eq. (3), entry i maximizes attention. The identity

$$\|q - k_i\|_2^2 - \|k_i\|_2^2 = \|q\|_2^2 - 2q^\top k_i$$

shows that these are power cells with sites k_i and weights $w_i = \|k_i\|_2^2$ (Aurenhammer, 1987). Equal-norm keys give the ordinary Voronoi case. Equivalently, the cells are the normal cones of the key convex hull; only its exposed vertices have full-dimensional retrieval cells, assuming distinct keys.

We use the *weighted Delaunay adjacency graph* whose vertices are full-dimensional cells and whose edges connect cells sharing a facet. An edge identifies two keys that exchange the winning role across a retrieval boundary. This definition leaves degeneracies untriangulated: the dot-product cells are cones meeting at the origin, so an arbitrary Delaunay triangulation would introduce unnecessary choices. MEMENTO does not construct this graph. Its scores select entries witnessed by rereading, thereby retaining a subset of its retrieval regions. Entries with lower-dimensional cells may still receive softmax mass and are not assumed harmless to remove.

B.3 Rereading as Geometric Witnessing

In the score of Eq. (4), the reference cache and the reread support $\mathcal{J}_r$ may differ, but shared original keys retain their representations. For grouped-query attention, r also ranges over the query heads sharing the KV head.

The inclusion $W_\tau \subseteq S_\tau$ in Eq. (5) is immediate: if some $r \in \mathcal{R}_i$ has $p_r(i) \geq \tau$, then $s_i = \max_r p_r(i) \geq \tau$. Every sufficiently strong cell witness therefore ensures retention. The maximum in the score makes one such witness sufficient, irrespective of how often other reread positions visit the cell. This viewpoint is related to geometric witnessing (de Silva & Carlsson, 2004), without asserting that rereading reconstructs a Delaunay complex. The following lemma gives a checkable condition under which a reread query is a strong witness.

Lemma 1 (A margin certifies a retained witness)**.** *Suppose $q_r^{\mathrm{rep}} \in \mathcal{C}_i$, $r \in \mathcal{R}_i$, and $\ell_i(q_r^{\mathrm{rep}}) - \ell_j(q_r^{\mathrm{rep}}) \geq \eta$ for every $j \in \mathcal{J}_r \setminus \{i\}$, where $\eta \geq 0$. Writing $n_r = |\mathcal{J}_r|$, we have*

$$s_i \geq p_r(i) \geq \frac{1}{1 + (n_r - 1)e^{-\eta}}.$$

Thus $i \in W_\tau$ whenever τ does not exceed this lower bound.

Proof. Divide numerator and denominator of $p_r(i)$ by $\exp(\ell_i(q_r^{\mathrm{rep}}))$ and bound each of the other $n_r - 1$ terms by $e^{-\eta}$. □

The margin condition includes auxiliary keys. A winning key among the original entries alone need not have a high reread probability, because auxiliary entries also contribute to the denominator.

B.4 Stability on Witnessed Cells

A future query in the covered region $\Omega_{\tau,\gamma}$ of Eq. (6) need not be close to its reread witness; it only needs to visit the same retained cell with adequate margin. The margin accounts for softmax contributions from competing keys, including those near Delaunay boundaries.

Proof of Proposition 1. For $q \in \Omega_{\tau,\gamma}$, some $i^* \in W_\tau \subseteq S$ has a logit at least γ above every competitor. It is therefore the unique maximizer both before and after pruning. Set $Z_E = \sum_{j \in E} e^{\ell_j(q)}$ and $Z_S = \sum_{j \in S} e^{\ell_j(q)}$. The margin gives

$$\frac{Z_E}{Z_S} \leq m e^{-\gamma}, \qquad \mu_E := \sum_{j \in E} \alpha_j(q) = \frac{Z_E}{Z_S + Z_E} \leq \frac{m e^{-\gamma}}{1 + m e^{-\gamma}}.$$

If E is nonempty, let $\bar{v}_E$ and $\bar{v}_S$ be the normalized attention-weighted means on the removed and retained sets. Then $o = \mu_E \bar{v}_E + (1 - \mu_E)\bar{v}_S$ and $o_S = \bar{v}_S$, so $\|o - o_S\|_2 = \mu_E \|\bar{v}_E - \bar{v}_S\|_2 \leq 2V\mu_E$. If E is empty the error is zero. This proves Eq. (7). Outside $\Omega_{\tau,\gamma}$, both outputs remain convex combinations of values of norm at most V, giving the universal bound $2V$. Splitting the expectation over the covered region and its complement proves Eq. (8). □

Size of the required margin. The bound separates missing geometric coverage, $\delta_{\tau,\gamma}$, from the residual attention tail inside covered cells. A substantial margin is necessary when many entries are removed: for $m > 0$ and $0 < \varepsilon < 1$, it suffices that $\gamma \geq \log(m(1 - \varepsilon)/\varepsilon)$ to make the tail at most ε. Simply preserving the hard-attention winner does not control diffuse softmax attention near cell boundaries.

Corollary 1 (Query-agnostic reuse over a covered domain)**.** *If a future-query domain $\mathcal{Q}$ satisfies $\mathcal{Q} \subseteq \Omega_{\tau,\gamma}$, then Eq. (7) holds uniformly over $\mathcal{Q}$ without using a future query to select S.*

Proof. $S = S_\tau$ is computed from reread queries only, and Eq. (7) holds pointwise on $\Omega_{\tau,\gamma} \supseteq \mathcal{Q}$. □

This supplies a geometric sufficient condition for reuse across turns: different requests may activate the same reread-witnessed cells. Coverage of the future domain is an explicit assumption; repeating token IDs alone does not establish it. In a streaming cache, each insertion or eviction changes the cell geometry. The result applies to each reference cache separately, with its own covered region, and

does not assert unchanged coverage across all subsequent cache states. It bounds a local attention operation with fixed representations, rather than accumulated changes to a complete autoregressive trajectory.

B.5 Proof and Discussion of the Main Theorem

The comparison in Theorem 1 holds the candidate KV representations and retained cardinality fixed, isolating selection from subsequent trajectory changes. The query-dependent policy selects S_D using current-turn cues in the history H_t; it does not see the future query drawn from $\mathcal{D}_t$.

Proof of Theorem 1. Applying Eq. (8) of Proposition 1 with $S = S_R$ and $m = |I| - |S_R|$ gives the upper bound $\mathcal{L}_t(S_R) \le U_R$. On the specified future-query event,

$$\|o(q) - v_i\|_2 = \left\| \sum_{j \neq i} \alpha_j(q)(v_j - v_i) \right\|_2 \le 2V(1 - \alpha_i(q)) \le 2V\varepsilon_D.$$

Since $o_{S_D}(q)$ belongs to the convex hull of retained values,

$$\|o(q) - o_{S_D}(q)\|_2 \ge \operatorname{dist}(v_i, \operatorname{conv}\{v_j : j \in S_D\}) - \|o(q) - v_i\|_2 \ge \Delta - 2V\varepsilon_D.$$

Nonnegativity of the error and the event probability yield $\mathcal{L}_t(S_D) \ge L_D$. Comparing this lower bound with the reread upper bound proves Eq. (10). □

Scope of the theorem. The term π captures future demand for entries lost under current-query selection; Δ prevents that demand from being met by redundant retained values. The reread bound depends on the probability mass of its covered cells, not their count. The theorem is conditional: max-reread scores do not automatically minimize δ_R, and a current query that accurately predicts future needs can favor query-dependent selection. Thus the claim is robustness to a specified shift in retrieval needs, not dominance for every future-query distribution.

B.6 Why Repeating Selection Each Turn Does Not Repair Lost Memory

The last inclusion of Eq. (11) follows because an old discarded identity belongs to neither C_t nor A_{t+1}. Re-evaluating the surviving entries under the next query therefore cannot undo earlier eviction. Newly arriving content can still supply equivalent information; value separation in Eq. (9) explicitly rules out that repair when applying the lower bound.

Theorem 1 applies conditionally at any turn where its assumptions hold. Over a shared sequence of reference histories, if the conditions hold at each turn with bounds $U_{R,t}$ and $L_{D,t}$, linearity of expectation gives

$$\mathbb{E} \sum_t \left[\mathcal{L}_t(S_{D,t}) - \mathcal{L}_t(S_{R,t}) \right] \ge \mathbb{E} \sum_t (L_{D,t} - U_{R,t}). \tag{12}$$

This compares cumulative local reuse error. It does not assert monotonic error growth or compare independently evolving agent trajectories, whose future queries and tool results can differ. Repeated query-dependent selection can make an early retrieval preference persistent through Eq. (11); rereading aims to make the selection before that irreversible step better suited to future reuse.

C Additional Experimental Details and Results

This appendix gives the details and results that support Section 4 but are not needed to follow it.

C.1 Full Experimental Setup

Models. Muse-Glimmer-30B (Meta Superintelligence Lab, 2026) has 13 full-attention and 39 sliding-window layers (window 2048) with 2 KV heads; Qwen3.8-27B (Qwen Team, 2026) has 16 full-attention and 48 linear-attention (Gated DeltaNet) layers with 4 KV heads; Qwen3-8B (Yang et al., 2025) has 36 full-attention layers with 8 KV heads. Only the full-attention layers hold a cache that grows with the session and only they are scored and evicted ($L = 13, 16, 36$); all runs are bf16

on NVIDIA B200 GPUs, with HuggingFace Transformers 4.51 for Qwen3-8B and 5.x for the other two.

Benchmarks. Table 6 summarizes the three benchmarks and their harnesses; they differ in what enters the context at each turn and in where the information that decides the task sits.

BrowseComp-Plus. BrowseComp-Plus (Chen et al., 2025) is a search agent over a fixed corpus of 100,195 documents; we use a fixed random subset of 200 of its 830 queries. The agent's only tool, `search`, returns the top 5 documents under a Qwen3-Embedding-8B retriever, each cut to its first 512 tokens. The question and the required answer format sit in a first prompt of 0.4k to 0.8k tokens; the evidence arrives in the documents retrieved over the session. A Qwen3-32B judge (Yang et al., 2025) (thinking off, temperature 0.7, top-p 0.8, top-k 20) grades the final answer against the gold answer with the grading prompt of the official BrowseComp-Plus evaluation script, and a query that ends without an answer counts as wrong.

τ^2-bench. τ^2-bench (Barres et al., 2025), the successor of τ-bench (Yao et al., 2024), has the agent serve a customer simulated by GPT-4.1 (`gpt-4.1-2025-04-14`, temperature 0) under a policy document through database tools. We run retail (all 114 tasks), telecom (the 114 base tasks under the workflow policy; dual control, so the user also acts on their own device) and banking (all 97 tasks; the task's reference documents are inlined into the system prompt instead of a retrieval tool). The rules sit in the first prompt and are present at prefill; each later turn adds a user message or the results of the agent's tool calls. One trial per task is scored as pass[1]: the reward is 1 only if every check in the task's reward basis holds, which is the final database state (retail, 88 banking tasks) or device state (telecom), the required actions (20 telecom and 9 banking tasks) and, on 40 retail tasks, assertions judged by GPT-4.1; a task that reaches 200 steps or 10 tool errors scores 0.

SWE-bench Verified. SWE-bench Verified (Jimenez et al., 2024) runs mini-SWE-agent (Yang et al., 2024) with one tool, `bash`, in the instance's container image, fixed by image ID (60 s per command; outputs longer than 10k characters are cut to their head and tail). Muse-Glimmer-30B calls the tool through its native tool-call format; Qwen3-8B writes each command in a fenced code block (mini-SWE-agent's text mode), because in tool-call mode 7 of its first 13 replies were malformed. The agent submits a `git diff`, which the official harness tests. A session also ends, without a patch and so unresolved, after 250 LLM calls, after three consecutive malformed replies, at the 131,072-position context limit, or after 3 h (2 h for Qwen3-8B). Here the model's own output is the largest part of the context: 42% of the final context, against 19% on BrowseComp-Plus and 27% to 40% on τ^2-bench for the same model, and nearly all the rest is the output of commands it chose.

Metrics. All three harnesses keep the history raw: every generated token, reasoning included, stays in the context for the rest of the session and is never stripped or re-rendered, so its cache entries leave only when the eviction method evicts them. Next to accuracy we report KV/head, the mean number of entries per KV head on the compressed layers, at the end of each query on BrowseComp-Plus and averaged over the agent's calls on τ^2-bench. Under eviction it sits slightly above M, because entries accumulate between evictions, which run every B tokens, and below M on a few short Qwen3-8B BrowseComp-Plus sessions that never fill the budget. CI is the 95% Wald half-width, and paired differences are tested with an exact two-sided McNemar test.

Agent loop and decoding. For Muse-Glimmer-30B and Qwen3.8-27B the harnesses bound the loop: after 25 searches on BrowseComp-Plus, or 25 tool calls on τ^2-bench (50 on banking), a further call is refused and the harness appends a directive to answer, which the model does without tools; the cap is checked once per generation, so a generation that chains several calls can overshoot it. A document or tool result already returned in the session is replaced by a short stub. Qwen3-8B runs without these limits, which it does not need: with the full KV cache it makes a median of 2 LLM calls and 1 search per BrowseComp-Plus query, 4 of its 200 queries exceed 25 searches, and 1 of its 325 τ^2-bench tasks exceeds the tool-call cap. Every agent samples with its model card's settings: temperature 1.0, top-p 0.95 and top-k 64 for Muse-Glimmer-30B, 1.0, 0.95 and 20 for Qwen3.8-27B, and 0.6, 0.95 and 20 for Qwen3-8B, with no repetition or presence penalty. Muse-Glimmer-30B reasons at strength *high*, Qwen3.8-27B thinks at its template's default effort (*xhigh*), and Qwen3-8B runs in thinking mode, with YaRN rope scaling ($\times 4$) on BrowseComp-Plus and SWE-bench Verified. Every run uses random seed 42, from which the harness derives the seed of each query, trial or instance.

Baselines. All four baselines evict in the same streaming regime at the same budget, on the same B-token clock, and keep the protected prefix P_{prot} of Table 6. StreamingLLM (Xiao et al., 2024b)

Table 6: Benchmarks and harness protocol. Under each benchmark the columns are Muse-Glimmer-30B, Qwen3.8-27B and Qwen3-8B; SWE-bench Verified is run on Muse-Glimmer-30B and Qwen3-8B, and ranges on τ^2-bench span its models and domains (for prompt size, its tasks).

	BrowseComp-Plus			τ^2-bench			SWE-bench Verified	
	30B	27B	8B	30B	27B	8B	30B	8B
Task								
Instances	200 of 830 queries			325 tasks: retail 114, telecom 114, banking 97			200 of 500, repository-stratified	
Agent's tool	`search` over 100,195 documents; top-5, 512 tokens each			domain database tools; simulated user (GPT-4.1)			`bash` in the instance container	
Input returned per turn	the 5 retrieved documents			a user message or a tool result			a command's output (file, search, test log)	
Deciding information	question in the prompt; evidence in documents retrieved later			policy, tool schemas and reference documents in the prompt (3.6k–15.3k tokens); request and database state later			issue in the prompt; code and test output read later	
Metric	accuracy, judged by Qwen3-32B			pass^1, one trial			resolved, official harness	
Session statistics (medians, full KV cache)								
LLM calls	27	18	2	11–19.5			74	21
Input per turn (tokens)	1.1k	1.3k	2.9k	46–220			126	140
Final context (tokens)	36k	43k	9.8k	11k–22k			35k	15k
Model-generated share	19%	39%	53%	27–46%			42%	49%
Harness limits								
Tool-call cap[†]	25	25	–	25[‡]	25[‡]	–	250 steps[¶]	
Max tokens per LLM call	16,384	32,768	8,192	6,144	16,384	6,144	8,192	8,192
Context limit (tokens)	126,000	150,576	126,000	131,072	131,072	40,960	131,072	131,072
MEMENTO *settings*								
Budget M (tokens)		4,096			4,096		8,192	
Protected prefix P_{prot}	prompt	prompt	4 sinks		4 sinks		prompt[§]	

[†] afterwards the agent must answer without tools; a dash means no cap. [‡] 50 on banking; 200 steps overall. [¶] the session then ends without a patch, as after 3 h (30B) or 2 h (8B). [§] truncated to $M-W-1$ tokens.

keeps the most recent entries after the protected ones; KeyDiff (Park et al., 2025), Expected Attention (Devoto et al., 2025) and LagKV (Liang et al., 2025) keep a recency window of 256. Expected Attention looks 512 positions ahead with the covariance and value-norm terms; LagKV scores with a lag of 1,024 and always keeps its first 16 entries and its last 1,024 to 2,047 entries (the last lag partition and the incomplete one after it). The four baselines keep exactly M entries per KV head; MEMENTO keeps MLH entries in total under one threshold, so for it M is an average per head.

Implementation details. MEMENTO runs Algorithms 2, 1 and 3 on HuggingFace Transformers with variable-length FlashAttention kernels, so the real and the re-fed token of a decoding step share one forward over one cache with different masks. Scores follow Eq. (1) and eviction Definition 3, with $M = 4096$, $B = 128$, $W = 256$ and $B_0 = 2048$ ($M = 8192$ on SWE-bench Verified); the first prompt fits one prefill chunk on BrowseComp-Plus and takes 2 to 8 chunks on τ^2-bench and 1 to 5 on SWE-bench Verified. The protected prefix P_{prot} follows Table 6.

C.2 Full BrowseComp-Plus Results

Table 7 extends Table 3 with the confidence interval, the share of answers in the required format, the number of searches and the final cache size.

Table 7: BrowseComp-Plus, 200 queries, $M = 4096$, $B = 128$, $W = 256$. CI: 95% Wald half-width (pp); Format: share of queries whose answer contains the required `Exact Answer:` field; Searches: executed search calls per query, capped at 25 for Muse-Glimmer-30B and Qwen3.8-27B; the cap is checked once per generation, and one Muse StreamingLLM generation chained 729 calls. KV/head: mean over queries. Bold: best eviction method.

Method	Acc. (%)	Correct	CI (pp)	Format (%)	Searches	KV/head
Muse-Glimmer-30B						
full KV cache	55.0	110/200	6.9	82.5	23.07	35,450
StreamingLLM	5.0	10/200	3.0	90.0	28.43	4,162
KeyDiff	11.5	23/200	4.4	25.0	24.77	4,168
Expected Attention	37.5	75/200	6.7	49.0	24.49	4,163
LagKV	9.0	18/200	4.0	84.0	24.78	4,160
MEMENTO	**51.0**	**102/200**	6.9	**93.5**	23.84	4,157
Qwen3.8-27B						
full KV cache	58.0	116/200	6.8	100.0	15.97	47,196
StreamingLLM	19.5	39/200	5.5	69.5	19.53	4,161
KeyDiff	8.0	16/200	3.8	79.5	19.46	4,166
Expected Attention	38.0	76/200	6.7	86.5	17.30	4,155
LagKV	28.5	57/200	6.3	**99.0**	18.54	4,163
MEMENTO	**48.5**	**97/200**	6.9	97.5	16.61	4,159
Qwen3-8B						
full KV cache	14.0	28/200	4.8	91.5	2.72	13,734
StreamingLLM	9.5	19/200	4.1	9.5	0.86	4,079
KeyDiff	7.5	15/200	3.7	**91.5**	1.80	4,088
Expected Attention	**12.5**	**25/200**	4.6	73.0	1.75	4,106
LagKV	11.0	22/200	4.3	70.0	1.75	4,079
MEMENTO	11.5	23/200	4.4	70.5	3.18	4,139

Table 8: τ^2-bench pass[1] (%) at $M = 4096$, $B = 128$, $W = 256$, one trial per task (114 / 114 / 97 tasks). KV/head, averaged over the agent's calls: 4,163 to 4,261 under eviction, 8.3k to 17.1k without. Bold: best eviction method per column.

	Muse-Glimmer-30B			Qwen3.8-27B			Qwen3-8B		
Method	retail	telecom	banking	retail	telecom	banking	retail	telecom	banking
full KV cache	83.3	92.1	43.3	86.0	93.0	42.3	41.2	18.4	5.2
StreamingLLM	36.8	23.7	5.2	7.0	14.9	7.2	13.2	15.8	**6.2**
KeyDiff	49.1	50.0	11.3	20.2	33.3	11.3	5.3	12.3	5.2
Expected Attention	21.9	22.8	9.3	6.1	34.2	13.4	17.5	**23.7**	4.1
LagKV	47.4	33.3	9.3	21.9	17.5	12.4	19.3	13.2	5.2
MEMENTO	**83.3**	**75.4**	**18.6**	**64.0**	**76.3**	**20.6**	**34.2**	16.7	3.1

MEMENTO keeps the instructions and the latest tool result accessible: on Muse-Glimmer-30B 93.5% of its answers follow the required format, against 49.0% for Expected Attention, and the agent exhausts the search cap on 88% of queries against about 96%. On Qwen3.8-27B 97.5% of its answers are in the required format. Every method keeps about M entries per head on average.

C.3 Full τ^2-bench Results

Table 8 extends Table 3 with Qwen3-8B.

On Qwen3-8B the full KV cache solves 41.2 / 18.4 / 5.2% and eviction 3.1 to 34.2%; with one trial per task we read no ranking from margins inside the confidence interval.

C.4 Full SWE-bench Verified Results

Table 9 extends Table 4 with Qwen3-8B and the sessions that end after three consecutive malformed replies.

Table 9: SWE-bench Verified, 200 instances, $M = 8192$, $B = 128$, $W = 256$, first prompt protected. Resolved: instances whose patch passes the official harness. Termination: how each session ended, by a submission, at the 131,072-position context limit, at the 250-step limit, or after three consecutive malformed replies; only a submission yields a patch. Steps: median LLM calls per session. Bold: best eviction method (fewest for the limits, malformed replies and steps). Qwen3-8B resolves at most 7 instances in any arm and no two of its arms differ (paired $p \geq 0.29$), so none of its cells is bold.

	Resolved		Termination (sessions)				
Method	#	%	Submitted	Ctx. limit	Step limit	Malformed	Steps
Muse-Glimmer-30B							
full KV cache	147	73.5	199	0	1	0	74
Expected Attention	130	65.0	164	10	26	**0**	87
MEMENTO	**143**	**71.5**	**189**	**1**	**10**	**0**	**74**
Qwen3-8B							
full KV cache	6	3.0	132	11	28	29	21
Expected Attention	7	3.5	144	7	33	16	20.5
MEMENTO	3	1.5	147	7	5	41	16

Table 10: **FastKVzip against MEMENTO**, same runs and settings as Tables 3 and 4: accuracy on BrowseComp-Plus and resolved rate on SWE-bench Verified (%). Full KV: the full KV cache. Bold: best eviction method per row.

Benchmark	Model	Full KV	FastKVzip	MEMENTO
BrowseComp-Plus	Muse-Glimmer-30B	55.0	7.0	**51.0**
	Qwen3.8-27B	58.0	16.5	**48.5**
SWE-bench Verified	Muse-Glimmer-30B	73.5	31.5	**71.5**

Qwen3-8B is at the floor of this benchmark: with the full KV cache it resolves 6/200 and leaves 128 of its 200 patches empty, and its three arms resolve 3 to 7 instances with no paired difference ($p \geq 0.29$), so we read no ranking from them.

C.5 Comparison with a Trained Scorer: FastKVzip

FastKVzip (Kim et al., 2026), the successor of KVzip (Kim et al., 2025), scores each entry with a gate trained to predict the attention that KVzip's context reconstruction pass would give it, so it needs no reconstruction pass at inference. Because the gate is trained, it is not in the class of the training-free scorers of Section 4, and we compare it with MEMENTO separately. It evicts in the same streaming regime, at the same budget and with the same protected prefix, with its own recency window of 128. Muse-Glimmer-30B and Qwen3.8-27B have no public gate, so we train gates for them with its recipe. Like MEMENTO, it keeps MLH entries in total under one threshold, so M is an average per head. Table 10 gives the results.

BrowseComp-Plus. FastKVzip is at the floor, 14/200 on Muse-Glimmer-30B and 33/200 on Qwen3.8-27B, below Expected Attention, with only 15.0% and 20.0% of its answers in the required format.

SWE-bench Verified. On Muse-Glimmer-30B FastKVzip resolves 63/200 (31.5%), against 143 for MEMENTO (3 wins to 83, $p < 0.001$). Few of its patches are wrong (7 of its 70 submissions are unresolved), but 118 of its 200 sessions end at the step or context limit, after a median of 193.5 calls against 74 for MEMENTO.

A trained gate thus falls behind a reread where the deciding content arrives during the session and must be re-measured under later cues.

C.6 Decoding Speed

Setup. The accuracy runs use HuggingFace Transformers (Appendix C.1). We measure speed separately, on our vLLM implementation of MEMENTO with Qwen3-8B on one NVIDIA H200 GPU, against full attention in the same engine. The workload follows the length profile of AIME24:

Table 11: **Which heads may score.** Muse-Glimmer-30B, default setting: correct BrowseComp-Plus queries (of 200) and solved τ^2-bench retail tasks (of 114), accuracy in parentheses, with the entry's own-reread credit only (no-refresh) and with the mean of all credits (MEMENTO). Bold: best arm per column; KV/head 4,160 to 4,230 in every arm.

	BrowseComp-Plus (/200)		τ^2-bench retail (/114)	
Scoring heads	no-refresh	mean	no-refresh	mean
Per head (MEMENTO)	24 (12.0)	**102** (51.0)	**100** (87.7)	**95** (83.3)
Retrieval heads (136)	**34** (17.0)	79 (39.5)	34 (29.8)	78 (68.4)
Complement (280)	16 (8.0)	89 (44.5)	33 (28.9)	78 (68.4)

each request has a 150-token prompt and generates 28,555 tokens, the longest AIME24 reference generation, so it ends at 28,705 tokens. MEMENTO uses $M = 4096$ and $W = 256$. We report decoding throughput in tokens/s and peak HBM, the NVML peak of the whole process tree, which includes every allocation of our implementation.

Results. The budget bounds how many entries each request stores and each step reads, so MEMENTO turns its memory savings into speed on long sequences. Under a fixed 24 GiB HBM budget, full attention holds one 28,705-token request (3.94 GiB of KV), while MEMENTO holds six (0.63 GiB each) and generates 574.5 against 181.5 tokens/s in aggregate, $3.17\times$ the throughput at nearly the same peak memory (23.8 vs 23.0 GiB); a single request alone is 8% slower ($0.92\times$). At a matched batch of four, with memory for both, MEMENTO uses 58% of the peak memory of full attention (22.2 vs 38.2 GiB) and decodes each request $1.02\times$ as fast over the whole generation. Per step it is slower at short contexts ($0.77\times$ from 4k to 8k tokens), even from 8k to 16k ($1.00\times$) and $1.18\times$ faster from 16k to 28.5k, since full attention reads every cached entry while MEMENTO's per-step cost, rereads included, is pinned by the budget. The gain is thus specific to long sessions: at 8k tokens both methods would fit six requests in 24 GiB, and the throughput advantage disappears.

C.7 Additional Ablations

Which heads may score. Under Definition 3 each KV head keeps the positions its own credits select. A needle-in-a-haystack copy test (Wu et al., 2025) qualifies 136 of the 416 query heads as retrieval heads; only they score and every other head keeps the positions they selected, or, as a control, the complementary 280 heads score instead. In Table 11 per-head decisions win three of the four columns, by 13 to 23 queries on BrowseComp-Plus under the mean rule ($p = 0.035$ and $p < 0.001$) and by 17 to 67 retail tasks ($p \leq 0.003$); the exception, no-refresh on BrowseComp-Plus, has every arm at the floor. Which heads score matters little: retrieval heads and complement are indistinguishable in three of the four comparisons ($p \geq 0.099$); the one gap ($p = 0.005$) is that floor cell. **The loss comes from imposing one shared set of positions on every head, not from picking the wrong heads:** the query that reaches an entry is head-specific, so a shared decision discards what the reread measured.

Prefill reread. Algorithm 2 rereads the first prompt causally, each B_0-token chunk's copy seeing only the chunks before it; our earlier runs reread it with one copy of the whole prompt, which coincides with it when the prompt fits one prefill chunk, as on BrowseComp-Plus. Under the mean rule on τ^2-bench, Muse is unchanged on banking (18 vs 18) and gains 16 telecom tasks with the causal reread (86 vs 70, $p = 0.007$); on Qwen3.8-27B the two arms are within noise ($p \geq 0.74$). The causal reread scores the start of the policy before the rest of the prompt exists, which helps when that start is what later turns recall (Muse telecom); we keep it for applying one rule to prompt and stream.

Sweep details. In Figure 2, the hollow markers at M=2048 are lower bounds: the protected prefix, the recency window and the unscored entries together exceeded the budget, so every query had evictions deferred. KV/head is within 2% of M except at B=256 and 512 (3% and 6–8% above), at M=2048 (6–7% above) and at M=24576 (3% below).

D Detailed Related Work

Query-dependent KV cache eviction. A large family of training-free methods bounds the cache by scoring entries with observed attention and discarding the rest. H2O keeps accumulated heavy

Table 12: **Prefill reread.** Solved τ^2-bench tasks (of 114 / 97), accuracy in parentheses, when each B_0-token chunk of the first prompt is reread causally (Algorithm 2) or with its copy placed after the whole prompt. Default setting, seed 42; mean rule, except Qwen3.8-27B (no-refresh). Bold: the better arm where paired McNemar $p < 0.05$.

Model	Prefill reread	telecom	banking
Muse-Glimmer-30B	causal	**86** (75.4)	18 (18.6)
	whole prompt	70 (61.4)	18 (18.6)
Qwen3.8-27B (no-refresh)	causal	95 (83.3)	21 (21.6)
	whole prompt	94 (82.5)	19 (19.6)

hitters (Zhang et al., 2023), Scissorhands exploits the persistence of importance across steps (Liu et al., 2023), and TOVA casts the bounded cache as a multi-state recurrent memory (Oren et al., 2024). SnapKV scores prompt entries with attention from a trailing observation window (Li et al., 2024), FastGen adapts the retention policy to per-head attention profiles (Ge et al., 2024a), PyramidKV varies the budget across layers (Cai et al., 2024), and Ada-KV allocates a global budget across heads (Feng et al., 2025). Query-aware sparsity methods such as Quest (Tang et al., 2024) and MInference (Jiang et al., 2024) keep the full cache and instead select which entries to attend to per query, so they avoid irreversible loss at the cost of retaining the memory. Our setting differs in what the score is conditioned on: these retention scores are read off the cues available at compression time, and when compression is repeated across turns, the surviving cache is progressively specialized to the requests already seen. Section 3.2 argues that later turns cannot recover from this by rescoring, and Eq. (11) states the corresponding inclusion.

Query-agnostic and attention-free retention scores. Several methods score entries without a task query. StreamingLLM retains attention sinks and a recent window (Xiao et al., 2024b), a position-based rule that is query-agnostic by construction but insensitive to content. KeyDiff scores keys by their geometric distinctiveness rather than by attention, which also keeps the method compatible with fused attention kernels (Park et al., 2025), and LagKV compares KV entries against lag-relative statistics within the cache itself (Liang et al., 2025). Expected Attention instead estimates, in closed form, how a distribution of future queries would attend to each entry (Devoto et al., 2025). Closest to our work, KVzip issues a repeat instruction and teacher-forces the context, scoring the original entries by reconstruction attention, which makes a single compressed cache reusable across many later queries (Kim et al., 2025); Fast KVzip removes the extra reconstruction pass by training a gating mechanism that predicts eviction directly (Kim et al., 2026). RazorAttention (Tang et al., 2025) and DuoAttention (Xiao et al., 2025) take a complementary route and assign budgets by head role, preserving the full cache only for retrieval heads. MEMENTO shares the query-agnostic objective of this line, and shares with KVzip the idea of scoring entries with model-generated attention rather than a heuristic statistic. It differs in two respects. First, the reread is the chunk's own content re-fed without a reconstruction instruction, so each entry is scored by the cue it would itself supply. Second, scoring is incremental and interleaved with decoding, so newly generated reasoning and incoming tool output are scored as they are produced, instead of compressing a supplied context once. Where Fast KVzip removes the cost of the reconstruction pass by learning a gate, MEMENTO keeps the training-free reread and removes its serial cost by scheduling, which leaves the score interpretable as attention from a witness query (Section 3.2). The name should not be confused with RE2 (Xu et al., 2023), which has the model read the input question twice to improve its answer; MEMENTO rereads its own context to decide what to keep, and adds nothing to the prompt.

Cache reuse across turns and serving systems. Systems work treats the cache as a reusable object: PagedAttention manages it in non-contiguous blocks (Kwon et al., 2023), SGLang reuses shared prefixes through a radix-structured cache (Zheng et al., 2024), Prompt Cache reuses attention states of modular prompt segments (Gim et al., 2024), and CacheBlend fuses precomputed caches of retrieved chunks (Yao et al., 2025). These systems decide what to keep or reuse at the level of text segments and do not score individual entries for future accessibility. A second line retains all entries but retrieves a subset per step, including landmark tokens (Mohtashami & Jaggi, 2023) and block-level context memory (Xiao et al., 2024a). Orthogonally, KIVI reduces bytes per entry by quantization (Liu et al., 2024b), and prompt or context compression shortens the input itself (Jiang et al., 2023; Mu et al., 2023; Ge et al., 2024b), which discards surface form rather than deciding which cached

entries survive. Agentic frameworks interleave reasoning, tool calls, and observations (Yao et al., 2023; Shinn et al., 2023) and manage context by paging text between tiers (Packer et al., 2023); the turn structure and tool-driven information needs of such interactions (Yao et al., 2024; Jimenez et al., 2024) are what make repeated, query-dependent compression of a persistent cache the failure mode we study.

Geometry and mechanism of attention retrieval. Our analysis reads keys as sites of a power diagram (Aurenhammer, 1987), which connects retrieval cells to weighted Delaunay adjacency and, in the approximate-nearest-neighbor literature, to proximity graphs built on the same dual (Malkov & Yashunin, 2020). Treating a re-fed query as a witness of the cell it activates parallels witness constructions in topological estimation (de Silva & Carlsson, 2004), though we do not claim that rereading recovers a Delaunay complex. On the mechanism side, induction heads show that a repeated prefix can retrieve its earlier continuation (Olsson et al., 2022), retrieval heads localize long-context factual recall to a small set of heads (Wu et al., 2025), and position effects in long contexts show that availability alone does not imply use (Liu et al., 2024a). These results motivate rereading as a test of accessibility, and we treat the question of which heads produce our scores as empirical.