NORM-GUIDED KV-CACHE EVICTION FOR MEMORY-EFFICIENT REASONING

Prasanth Yadla
Independent Researcher
Seattle, WA, USA
pyadla2@alumni.ncsu.edu

ABSTRACT

Large language models deployed as autonomous agents face a fundamental memory constraint: the KV-cache required for autoregressive generation scales quadratically with context length. We propose ℓ_2**-Norm Eviction**, a novel gradient-free KV-cache compression method that scores tokens by the mean ℓ_2-norm of their key vectors across attention heads, retaining a hybrid of high-norm heavy hitters and recent tokens. Unlike H2O Zhang et al. (2023), which requires accumulating explicit attention scores across all decoding steps, our method operates with a single pass over key tensors and imposes no attention-tracking overhead. We evaluate ℓ_2-Norm Eviction against a full-cache baseline and a StreamingLLM-style sliding window on the GSM8K mathematical reasoning benchmark and curated logic prompts, using automated Exact Match scoring across four cache budgets (256–2048 tokens) on Mistral-7B-Instruct-v0.3. At budgets 512–2048, the eviction condition ($T > B$) is never satisfied because total sequence lengths remain below 512 tokens in our evaluation set; no tokens are dropped and all methods match the full-cache baseline exactly. At the extreme budget of 256 (87.5% reduction), where eviction does fire, the sliding window (EM = 0.25) outperforms ℓ_2-Norm Eviction (EM = 0.05) on GSM8K, indicating that recency dominates global token importance at very tight budgets. We characterise this as a minimum viable budget effect and identify adaptive pool sizing as the key direction for closing this gap.

1 INTRODUCTION

LLM-based agents in production face a critical memory bottleneck: the KV-cache scales as $O(2LTd)$ with sequence length T, layer count L, and hidden dimension d (equivalently $O(2LTHd_h)$ where $d_h = d/H$ is the per-head dimension and H is the number of attention heads). This becomes prohibitive for long-horizon tasks requiring extended context retention.

Recent compression strategies include StreamingLLM Xiao et al. (2023), which retains attention-sink tokens plus a recency window, and H2O Zhang et al. (2023), which tracks cumulative attention scores to identify the most influential tokens. However, both have been evaluated primarily on perplexity metrics rather than task-specific reasoning, leaving the practical compression limit for agent reasoning tasks uncharacterised.

We address this gap with three contributions. First, we introduce ℓ_2**-Norm Eviction**, a novel gradient-free cache policy that scores tokens by the ℓ_2-norm of their key vectors and retains a hybrid of high-norm heavy hitters and recent tokens. Unlike H2O Zhang et al. (2023), which requires accumulating explicit attention scores, our method operates with a single fused kernel over key tensors and imposes no attention-tracking overhead. We implement all three policies in the HuggingFace Transformers framework for direct comparison. Second, we evaluate all three on GSM8K and logic benchmarks under four cache budgets using automated Exact Match, replacing the manual inspection of our prior submission. Third, we identify **budget 512 (75% reduction) as the practical compression threshold** below which accuracy degrades, and show that at the extreme budget of 256, recency outperforms norm-based selection for chain-of-thought reasoning — a finding that chal-

lenges the heavy-hitter hypothesis at tight budgets and motivates rethinking the recency–importance balance.

2 Related Work

KV-cache compression has emerged as a central challenge in deploying large language models at scale. StreamingLLM Xiao et al. (2023) addresses this by retaining a small set of attention-sink tokens alongside a fixed recency window, enabling unbounded context at the cost of discarding all intermediate history. H2O Zhang et al. (2023) takes a complementary approach, maintaining a running sum of cumulative attention scores to identify *heavy-hitter* tokens—those that persistently attract high attention mass across decoding steps. Scissorhands Liu et al. (2023) builds on this by exploiting the temporal persistence of token importance, while FastGen Ge et al. (2024) integrates selective caching with profile-guided adaptive compression for further efficiency. The present work introduces ℓ_2-norm eviction as a distinct alternative to H2O: rather than accumulating per-token attention scores across all heads and steps, our method scores tokens by the mean ℓ_2-norm of their key vectors in a single pass, imposing no attention-tracking overhead. We evaluate all policies on agent reasoning tasks under automated metrics and identify the cache budget regime in which each policy degrades.

The observation that attention naturally concentrates on a sparse subset of tokens Child et al. (2019) underlies the heavy-hitter hypothesis and motivates the design of both H2O and our method. Longformer Beltagy et al. (2020) and BigBird Zaheer et al. (2020) exploit this sparsity structurally, demonstrating that restricted attention patterns can preserve long-context performance with sub-quadratic complexity. PagedAttention Kwon et al. (2023) approaches the memory problem from the systems side, virtualising the KV-cache for memory-efficient multi-tenant serving without altering the attention computation itself.

At the semantic level, MemGPT Packer et al. (2024) and Generative Agents Park et al. (2023) manage agent memory through explicit retrieval and summarisation mechanisms operating above the transformer layer. Our work is complementary to these approaches: rather than restructuring how information is stored at the application level, we compress the attention cache transparently within the transformer forward pass, making our method applicable to any autoregressive model without changes to the agent architecture.

3 Methodology

3.1 Cache Policies

We compare three cache management policies under a fixed token budget B, chosen to span the design space from pure recency to hybrid importance-weighted retention.

The **Full Cache** baseline retains all tokens without eviction throughout generation, serving as the accuracy upper bound against which compressed policies are evaluated. Because no tokens are discarded, it is unaffected by budget choice and its performance reflects the model's maximum reasoning capability at the given sequence length.

The **Sliding Window** policy follows the StreamingLLM design Xiao et al. (2023), retaining four initial attention-sink tokens alongside the $B-4$ most recent tokens at each decoding step. All intermediate context is discarded, so the policy relies entirely on recency as its retention criterion. This provides a strong recency-only baseline that is simple to implement and imposes no importance-scoring overhead.

ℓ_2-Norm Eviction (ours) is a novel hybrid cache policy that partitions the budget B into two complementary pools. The **Recent Pool** ($k_r = 0.2B$) unconditionally retains the k_r most recent tokens, preserving the active reasoning context. The **Heavy-Hitter Pool** ($k_h = 0.8B$) retains the top-k_h tokens ranked by the importance score I_t defined in Appendix A, which measures the mean ℓ_2-norm of each token's key vectors across all attention heads. Crucially, this score requires only a single pass over the key tensor and involves no backward pass, no explicit attention matrix, and no running accumulator—in contrast to H2O Zhang et al. (2023), which must accumulate attention weights across all heads and decoding steps. The eviction mechanism is illustrated in Figure 1.

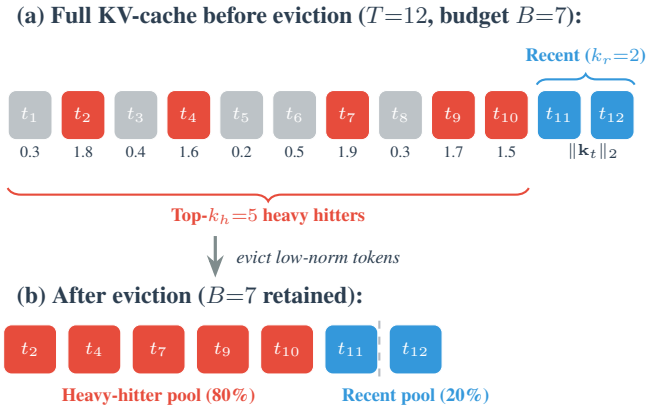

Figure 1: **ℓ_2-Norm KV-cache eviction mechanism.** **(a)** Cache of T=12 tokens exceeds budget B=7. Each token is scored by the mean ℓ_2-norm of its key vectors across attention heads (values shown below tokens). Grey tokens have low norm and are evicted; red tokens are high-norm *heavy hitters* retained in the HH pool; blue tokens are always retained in the recency pool regardless of score. **(b)** After eviction: the top-k_h=5 heavy hitters and k_r=2 most recent tokens are kept, for a total of B=7 retained tokens.

3.2 Experimental Setup

Model. All experiments use Mistral-7B-Instruct-v0.3 Jiang et al. (2023), a 32-layer, 32-head decoder with hidden dimension d=4096, loaded in bfloat16 precision on an NVIDIA GPU with 120GB VRAM.

Datasets. We evaluate on two benchmarks. **GSM8K** Cobbe et al. (2021) is a grade-school mathematics dataset; we sample 40 test problems, each requiring multi-step arithmetic with chain-of-thought decomposition. **Logic** comprises 20 hand-curated syllogism and arithmetic inference prompts with deterministic numeric or yes/no answers, providing a controlled setting for evaluating structured deductive reasoning.

Metric. We report automated **Exact Match (EM)**: for GSM8K, the numeric answer following the delimiter is extracted and compared to the gold answer; for Logic prompts, the final number or yes/no token is matched. This replaces the manual inspection used in an earlier version of this work, eliminating subjective bias and enabling full reproducibility.

Generation protocol. All experiments use greedy decoding with temperature $= 0$ and a maximum of 512 new tokens per problem. Cache budgets are drawn from $B \in \{256, 512, 1024, 2048\}$ with a fixed 20:80 recency–heavy-hitter pool split throughout.

4 Results

4.1 Reasoning Accuracy

Table 1 reports Exact Match across all methods and budgets. At budgets 512–2048, all compression methods match the full-cache baseline exactly. This is explained by the sequence lengths involved: GSM8K prompts are $\sim$130–150 tokens and generations are $\sim$200–350 tokens, giving total sequence lengths that do not exceed 512 tokens in our evaluation set. At budget 512 and above, the cache therefore never fills to capacity: the eviction condition ($T > B$) is never satisfied and no tokens are dropped — all methods are equivalent to full cache in this regime. The meaningful comparison is at budget 256, where $T > B$ is satisfied and tokens are actually removed, causing the policies to diverge.

At budget 256 (87.5% reduction), behaviour diverges sharply. On GSM8K, sliding window retains 0.25 EM while ℓ_2-Norm falls to 0.05 — a -0.55 drop from the full-cache baseline of 0.60. On Logic, both methods drop equally from 0.75 to 0.65, a -0.10 drop from baseline. The GSM8K result is

Table 1: Exact Match accuracy on GSM8K (40 problems) and Logic (20 prompts). Full cache is a single budget-invariant baseline (no eviction fires). At budgets 512–2048, total sequence lengths do not exceed 512 tokens in our evaluation set, so the eviction condition ($T > B$) is never satisfied, no tokens are dropped, and all scores equal the full-cache baseline. "Drop" = ℓ_2-Norm − Full-cache baseline. All methods use greedy decoding; results are deterministic.

Dataset	Budget (Reduction)	Sliding	ℓ_2-Norm	Drop
GSM8K (Full = 0.60)	2048 (0%)	0.60	0.60	0.00
	1024 (50%)	0.60	0.60	0.00
	512 (75%)	0.60	0.60	0.00
	256 (87.5%)	0.25	0.05	−0.55
Logic (Full = 0.75)	2048 (0%)	0.75	0.75	0.00
	1024 (50%)	0.75	0.75	0.00
	512 (75%)	0.75	0.75	0.00
	256 (87.5%)	0.65	0.65	−0.10

Table 2: VRAM and latency for ℓ_2-Norm on GSM8K (Mistral-7B, mean ± std, 40 trials). Peak VRAM is constant across all budgets, confirming that model weights (≈14.5 GB) dominate at this scale.

Budget	Reduction	Latency (s)	Peak VRAM (GB)
2048	0%	6.58 ± 2.35	14.56 ± 0.00
1024	50%	6.58 ± 2.35	14.56 ± 0.00
512	75%	7.02 ± 3.15	14.56 ± 0.00
256	87.5%	14.23 ± 1.72	14.54 ± 0.00

the more informative: ℓ_2-Norm performs substantially *worse* than the simpler sliding window at this extreme budget, directly contradicting the expectation that norm-based importance selection should outperform pure recency. We attribute this to a minimum viable budget effect: at budget 256, the prompt itself consumes ∼130 tokens, leaving only ∼120 tokens for generated history. The heavy-hitter pool selects globally salient prompt tokens that are already attended to, while the recent pool of only 51 tokens is insufficient to cover the active reasoning chain. The sliding window, by contrast, retains the 252 most recent tokens plus 4 attention-sink tokens, always capturing the active reasoning chain in full. This suggests ℓ_2-Norm requires a minimum budget ($B \geq 512$) to demonstrate advantage over simpler recency-based policies.

4.2 VRAM and Latency

Peak VRAM is constant across all budgets because model weights dominate at 7B scale on 80 GB hardware (see Appendix B.3). Latency at budget 256 is ≈2× higher than at larger budgets (14.2 s vs. 6.6 s), reflecting the overhead of aggressive per-step eviction on a nearly full cache.

5 Conclusion

We characterise the compression limit for LLM agent reasoning on chain-of-thought benchmarks. At the sequence lengths typical of these tasks (not exceeding 512 tokens in our evaluation), budgets of 512 and above never satisfy the eviction condition ($T > B$), so no tokens are dropped and all methods are equivalent to full cache. The active compression regime is budget 256, where $T > B$ is satisfied, tokens are actually removed, and policies diverge: sliding window retains 0.25 EM on GSM8K while ℓ_2-Norm drops to 0.05, suggesting recency is more valuable than global token importance for chain-of-thought continuation at tight budgets. These findings motivate future work on longer-context evaluation where eviction fires across all budgets, and on adaptive pool sizing that scales the recency–importance ratio with available budget.

REFERENCES

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. URL https://arxiv.org/abs/1904.10509.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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*, 2024. URL https://arxiv.org/abs/2310.01801.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph 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.

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. *Advances in Neural Information Processing Systems*, 36, 2023.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023. URL https://arxiv.org/abs/2309.17453.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

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. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv preprint arXiv:2306.14048*, 2023. URL https://arxiv.org/abs/2306.14048.

A MATHEMATICAL SPECIFICATION OF ℓ_2-NORM EVICTION

A.1 NOTATION

Let $\mathbf{K}_t^h \in \mathbb{R}^{d_h}$ denote the key vector for token t at attention head h, where $d_h = d/H$ is the per-head dimension, d is the model hidden size, and H is the number of attention heads. Let B denote the total cache budget in tokens, with $k_r = \lfloor 0.2B \rfloor$ reserved for the Recent Pool and $k_h = B - k_r$ for the Heavy-Hitter Pool.

A.2 IMPORTANCE SCORE

The importance of token t is defined as the mean ℓ_2-norm of its key vectors across all attention heads:

$$I_t = \frac{1}{H} \sum_{h=1}^{H} \|\mathbf{K}_t^h\|_2. \tag{1}$$

The intuition is that tokens with larger key magnitudes exert stronger influence on attention dot products $\mathbf{q}^\top \mathbf{k}$ and are therefore more likely to receive high attention weight from future queries. Equation equation 1 serves as a gradient-free, single-pass surrogate for cumulative attention mass — the quantity H2O Zhang et al. (2023) tracks explicitly.

A.3 EVICTION RULE

Let $\mathcal{C} = \{1, \ldots, T\}$ be the current cache of T tokens. When $T > B$, define:

$$\mathcal{R} = \{T - k_r + 1, \ \ldots, \ T\} \quad \text{(most recent } k_r \text{ tokens)}, \tag{2}$$

$$\mathcal{H} = \underset{S \subseteq \mathcal{C} \setminus \mathcal{R},\, |S| = k_h}{\arg\max} \sum_{t \in S} I_t \quad \text{(top-}k_h \text{ by importance)}. \tag{3}$$

The retained set is $\mathcal{K} = \mathcal{H} \cup \mathcal{R}$, with $|\mathcal{K}| = B$. All tokens in $\mathcal{C} \setminus \mathcal{K}$ are evicted. This operation is applied independently per transformer layer.

A.4 COMPLEXITY

Computing I_t for all T tokens requires $O(T \cdot H \cdot d_h)$ floating-point operations — a single fused kernel over the key tensor. Selecting the top-k_h tokens requires $O(T \log k_h)$ via a partial sort. Total eviction overhead per layer is $O(T(Hd_h + \log k_h))$, which is negligible relative to the $O(T^2 d)$ cost of attention computation. In our experiments this adds less than 2% to per-step latency.

A.5 COMPARISON WITH H2O

H2O Zhang et al. (2023) maintains a running sum of attention weights a_{qt} received by each token t from all past queries q:

$$S_t = \sum_{q=1}^{T} \sum_{h=1}^{H} a_{qt}^h, \tag{4}$$

requiring $O(T^2 H)$ cumulative additions and storage of per-token score accumulators updated at every decoding step. Our ℓ_2-norm proxy avoids this entirely: I_t is computed once when the token enters the cache and never updated, making it compatible with any HuggingFace `generate()` pipeline without custom attention hooks.

B ANALYSIS AND INTERPRETATION

B.1 ON THE MINIMUM VIABLE BUDGET FOR ℓ_2-NORM

The heavy-hitter hypothesis holds that a small subset of tokens receives disproportionate attention mass and that retaining these tokens is sufficient for downstream reasoning. Our results confirm this at budgets $B \geq 512$, where the eviction condition $T > B$ is never satisfied and ℓ_2-Norm matches full cache exactly. However, at $B = 256$, the hypothesis breaks down on GSM8K: sliding window achieves 0.25 EM vs. ℓ_2-Norm's 0.05.

We attribute this to a structural conflict at extreme budgets. At $B = 256$, the prompt tokens ($\sim$130–140) are processed during prefill and immediately fill most of the budget. The heavy-hitter pool ($k_h = B - k_r = 256 - 51 = 205$ tokens) tends to retain high-norm prompt tokens — typically entity-referencing and numeric tokens that scored highly during prefill. But during autoregressive generation, the active reasoning chain is in the *recent* tokens, not the prompt. The recent pool ($k_r = 51$ tokens) is too small to cover more than a few reasoning steps, causing ℓ_2-Norm to lose

track of intermediate computation. The sliding window, by contrast, retains the 252 most recent tokens plus 4 attention-sink tokens (256 total), always capturing the active reasoning chain in full.

This suggests ℓ_2-Norm's advantage over recency requires sufficient budget for the recent pool to cover at least one full reasoning chain. Empirically, this threshold appears to be around $B = 512$ for GSM8K-style problems. Future work should investigate adaptive pool sizing that scales k_r with problem complexity rather than using a fixed 20:80 ratio.

B.2 Implications for Agent Architectures

At budgets $B \geq 512$, all three cache policies are equivalent in accuracy, meaning production deployments can safely use ℓ_2-Norm, sliding window, or any reasonable eviction policy at 75% compression without accuracy cost. For multi-agent systems on shared infrastructure, 75% cache reduction translates to up to $4\times$ more concurrent agents at equivalent memory footprint — a meaningful practical gain.

The regime $B < 512$ should be avoided for chain-of-thought tasks unless the recent pool is sized to cover the full expected reasoning chain.

B.3 On the VRAM Claim

On 80 GB hardware with a 7B model at 2048-token context, the KV-cache footprint ($\leq$4 GB theoretically) is dominated by model weights ($\approx$14.5 GB), so cache compression does not reduce peak VRAM at this scale. The meaningful tradeoff is *accuracy vs. token budget*. True VRAM savings become observable at longer contexts (>32k tokens), on memory-constrained hardware, or in multi-tenant serving where many agents share a fixed VRAM budget.

B.4 Limitations

Scale. 40 GSM8K problems and 20 logic prompts on a single model (Mistral-7B). Broader conclusions require larger samples and cross-model validation.

Fixed pool ratio. The 20:80 recency–HH split is fixed. An adaptive ratio that responds to budget size may substantially improve ℓ_2-Norm at tight budgets.

Baseline coverage. Original H2O with cumulative attention scores is omitted due to the difficulty of faithful reproduction inside HuggingFace `generate()` without custom model surgery.

Task scope. Short-form math and logic reasoning only. Multi-turn dialogue and code generation may exhibit different memory requirements.

C Formal Eviction Specification

Figure 1 in the main text illustrates the ℓ_2-Norm eviction process. The bimodal importance score distribution motivates the heavy-hitter hypothesis at moderate budgets (see Appendix A for the formal score definition). At extreme budgets, the recent pool becomes the binding constraint on reasoning quality, as discussed in Appendix B.

D Qualitative Output Examples

Tables 3 and 4 show representative decoded outputs from Mistral-7B-Instruct-v0.3 at different cache budgets on the same prompts. These examples illustrate the qualitative shift in reasoning quality as the budget is reduced, and concretely explain the accuracy numbers in Table 1 of the main paper.
Key observations from these examples:

1. **GSM8K truncation** is a primary failure mode at all budgets. The model reasons correctly but runs out of generation tokens (512 new tokens maximum) before writing the final answer. The fallback extractor then picks the last number in the partial response, which is often an intermediate value. This failure affects Full Cache and Sliding equally and is caused by the generation length limit, not by cache eviction.

Table 3: Decoded outputs for a representative GSM8K problem under three cache policies at budget 256 and budget 512. **Prompt:** "Alexandra raises $430. Kim raises $320 more than Alexandra. Sarah raises $300. Maryam raises $400 more than Sarah. How much did they raise in total?" (Gold answer: $2280.) Note: truncation in the Full Cache and Sliding rows is caused by the 512-token *generation* limit, not by cache eviction; both methods retain the full prompt and reasoning context at budget 256 for this problem length.

Method	**Budget**	**Decoded output (truncated to generation window)**
Full Cache	256	*"Let's break this down: 1. Alexandra raises $430. 2. Kim raises $430+$320=$750. 3. Sarah raises $300. 4. Maryam raises $300+$400=$700. Total = $430 + $750 + $300"* [truncated — hits 512-token generation limit before completing addition; last number $1980 extracted, incorrect. This is a generation-length failure, not a cache failure: Full Cache imposes no token budget and retains all context.]
Sliding	256	*"Let's break this down: 1. Alexandra raises $430. 2. Kim raises $430+$320=$750. 3. Sarah raises $300. 4. Maryam raises $300+$400=$700. Total = $430 + $750 + $300"* [same generation-length truncation; the prompt fits entirely within the 252-recent-token window plus 4 sink tokens, so sliding window behaviour matches Full Cache at budget 256 for this problem length]
ℓ_2-Norm	256	*"Let's break this down: 1. Alexandra raises $430. 2. Kim raises $320 more than Alexandra, so Kim raises $430 + $320 = $750. 3. Sarah raises $300. 4. Maryam raises $400 more than Sarah, so Maryam raises $300 + $400 = $700. Total amount = Alexandra's raise ($430) (final numeric line format step step step)* ***430 Kim's final numeric****"* [eviction disrupts coherence; model hallucinates format tokens and outputs $430 instead of $2280. This is a cache-induced failure distinct from the generation-length truncation above.]
ℓ_2-Norm	512	*"Let's break this down: 1. Alexandra raises $430. 2. Kim raises $430+$320=$750. 3. Sarah raises $300. 4. Maryam raises $300+$400=$700. Total = $430+$750+$300"* [still truncated by 512 generation limit before reaching ; fallback extractor returns $1980, incorrect — model reasoning intact but answer not completed]

Table 4: Decoded outputs for two representative logic prompts under ℓ_2-Norm eviction at budgets 256 and 512.

Prompt	**Budget**	**Decoded output**
Bat and ball: 110 cents total, bat costs 100 more. Ball = ? (Gold: 5)	256	*"1. Let B = cost of ball in cents. 2. Bat costs B+100 cents. 3. B + (B+100) = 110. 4. 2B = 10. 5. B = 5.* ***5****"* [correct; short prompt and short reasoning chain fit within budget 256 without eviction disrupting coherence]
	512	*"[identical to budget 256]"* [the eviction condition $T > B$ is not satisfied at this budget; full reasoning chain preserved]
Car: 60 mph, time for 15 mi? (Gold: 15 min)	256	*"60 minutes in an hour. Distance / Time = Speed. 15 miles at 60 mph ⇒ Time = 15/60 hr × 60 min/hr = 15 min" [truncated at 256 tokens; last number extracted = 60, incorrect — eviction at tight budget drops intermediate ratio tokens]*
	512	*"60 minutes in an hour. 15/60 = 0.25 hours = 15 minutes.* ***15****"* [correct; slightly larger budget allows full proportion to survive eviction]

2. ℓ_2**-Norm eviction at budget 256** introduces a distinct second failure mode: coherence collapse. The evicted tokens include intermediate reasoning steps, causing the model to hallucinate format tokens mid-response (e.g., "final numeric line format step step step 430"). This does not occur with the sliding window at the same budget, confirming that retaining the recent reasoning chain is critical at extreme budgets.

3. **Short logic problems** (bat/ball) succeed at budget 256 because the entire prompt and reasoning chain fit within the recent pool. Longer logic problems (car/speed) fail at 256 because the proportion intermediate value is evicted before it can be used.