KVEraser: Learning to Steer KV Cache for Efficient Localized Context Erasing

Anonymous Authors[1]

Abstract

Context erasing for KV cache is challenging because a local edit has a global consequence: once a span has been processed, its influence propagates into the cached states of all subsequent tokens. Exact erasing must recompute all tokens after the deleted span. We introduce KVEraser, a learned KV-cache editing method for efficient localized context erasing. Experiments show that KVEraser matches full recomputation's performance on in-domain tasks across 1K–32K context lengths, while its latency increases by only 24% compared with a 17.6× increase for full recomputation. KVEraser also generalizes to unseen QA tasks with harmful factual distractors, achieving a 3–4× speedup over full recomputation.

1. Introduction

Key-Value (KV) caching is a core mechanism for efficient inference in large language models (LLMs). After context processing, the cached keys and values allow each subsequent token to attend to the previous context without recomputation. Modern serving systems rely heavily on KV-cache management (Kwon et al., 2023; Zheng et al., 2024).

In many long-context applications, the context processed by an LLM is assembled online from retrieved documents, tool observations, and execution logs (Lewis et al., 2020; Schick et al., 2023; Yao et al., 2023). A RAG system may prefill retrieved passages before discovering an out-of-date fact; a long-running agent may continue from a cached context before detecting adversarial instructions in an imported skill file (Shi et al., 2023; Xie et al., 2024; Sun et al., 2024; Greshake et al., 2023; Schmotz et al., 2025). In all these cases, a short problematic span is identified only after prefill, when the long context has already been written into the KV cache. Such stale, incorrect, or harmful context can lead to incorrect answers, failed actions, or unsafe behavior. The desired inference-time operation is therefore **context erasing**: given a processed context and a span to delete, make future decoding behave as if that span had never appeared.

The difficulty of context erasing is rooted in a strict validity condition of KV reuse. Exact KV reuse requires the same preceding context, i.e., prefix. Under causal self-attention, the representation of each token depends on all earlier tokens; once an earlier part of the context is edited, the cached states of all later tokens are no longer exact. The exact solution is to reuse the unchanged prefix cache and rerun prefill over the entire impacted suffix. Its cost is therefore governed by suffix length rather than deleted-span length, making it increasingly expensive as context sizes grow.

These observations motivate a **question**: can post-hoc context erasing be performed directly in KV space, without rerunning prefill on the suffix? To this end, we introduce **KVEraser**, a learned cache-editing method that generates steering KV states to replace the original KV states of the erased span while reusing the rest of the cache unchanged. The eraser module is trained to make decoding from this surrogate cache approximate the counterfactual behavior of the frozen model on the edited prompt with the span deleted.

On a controlled long-context erasing benchmark, KVEraser achieves near-perfect post-erasure performance across contexts from 1K to 32K tokens, matching full recomputation. Over the range, its latency increases by only 24%, compared with $17.6\times$ for full recomputation. On unseen question answering datasets, KVEraser achieves the best performance among approximate methods at comparable or lower latency, while full recomputation remains 3–4× slower.

2. Background and Problem Formulation

KV Cache. LLMs process a context $\mathbf{x} = (x_1, \ldots, x_T)$ with causal attention (Radford et al., 2018). For each token at position i, a transformer layer first computes query, key, and value vectors $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i$. Positional encoding is integrated here to preserve relative token ordering information. The model then performs attention-based aggregation $\sum_{1\le j\le i} w_{ij}\mathbf{v}_j$, where $\{w_{ij}\}_{1\le j\le i}$ is obtained by applying softmax to $\{\mathbf{q}_i^T\mathbf{k}_j\}_{1\le j\le i}$ (Vaswani et al., 2017). By caching previously computed key and value vectors

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by *The Impact of Memorization on Trustworthy Foundation Models* Workshop @ ICML. Do not distribute.

$\{\mathbf{k}_j\}_{1\le j<i}, \{\mathbf{v}_j\}_{1\le j<i}$, we avoid recomputing $\mathbf{k}_j$ and $\mathbf{v}_j$ for $j < i$ when new tokens are appended.

Context erasing asks a question: after a context has been prefilled, can we make the model behave as if a chosen span had never appeared? We study a clean and general setting in which a single contiguous span is deleted. We decompose the processed context as $\mathbf{x} = \mathbf{p} \oplus \mathbf{e} \oplus \mathbf{s}$, where $\mathbf{p} = \mathbf{x}_{1:m-1}$ is the prefix, $\mathbf{e} = \mathbf{x}_{m:n}$ is the span to erase, and $\mathbf{s} = \mathbf{x}_{n+1:T}$ is the suffix. The edited prompt after deleting $\mathbf{e}$ is $\tilde{\mathbf{x}} = \mathbf{p} \oplus \mathbf{s}$.

Let $\mathbf{KV}(\mathbf{x})$ denote the KV cache of a context $\mathbf{x}$. The cached state of each token depends on its entire prefix. Hence the KV cache of the first $m-1$ tokens remains valid after deleting $\mathbf{e}$, whereas the cache of the remaining $T-n$ tokens is contaminated by the erased span. Exact erasing could be performed by reusing the unchanged prefix cache and rerunning the prefill pass on the edited prompt $\tilde{\mathbf{x}}$, yielding $\mathbf{KV}(\tilde{\mathbf{x}})$. Our goal is to study whether context erasing can be performed without expensive full-suffix recomputation. A KV-cache reuse strategy for context erasing is a procedure

$$\pi : (\mathbf{x}, \mathbf{KV}(\mathbf{x}), m{:}n) \mapsto \widehat{\mathbf{KV}}(\mathbf{x}; m, n), \tag{1}$$

which constructs a surrogate KV cache for the edited prompt without re-prefill on $\tilde{\mathbf{x}}$. Given a user request $\mathbf{u}$, reuse is considered successful if:

$$p_\theta(\mathbf{a} \mid \mathbf{KV}(\tilde{\mathbf{x}}), \mathbf{u}) \;\approx\; p_\theta(\mathbf{a} \mid \widehat{\mathbf{KV}}(\mathbf{x}; m, n), \mathbf{u}). \tag{2}$$

3. KVEraser

3.1. Surrogate Cache Construction

Let $\mathbf{KV}_{a:b}(\mathbf{x})$ denote the part of $\mathbf{KV}(\mathbf{x})$ for positions $a, \dots, b$. The original cache decomposes as

$$\mathbf{KV}(\mathbf{x}) = \mathbf{KV}_{1:m-1}(\mathbf{x}) \oplus \mathbf{KV}_{m:n}(\mathbf{x}) \oplus \mathbf{KV}_{n+1:T}(\mathbf{x}).$$

Local Edits. The prefix cache $\mathbf{KV}_{1:m-1}(\mathbf{x})$ remains valid after deleting $\mathbf{e}$, so we keep it unchanged. We propose to reuse the suffix cache and replace only the erased interval with a learned, length-preserving steering block:

$$\widehat{\mathbf{KV}}(\mathbf{x}; m, n) = \mathbf{KV}_{1:m-1}(\mathbf{x}) \oplus \mathbf{KV}^{\text{steer}}(\mathbf{x}; m, n) \tag{3}$$

$$\oplus\, \mathbf{KV}_{n+1:T}(\mathbf{x}). \tag{4}$$

Why Can This Work? Consider one attention layer and focus on the contribution from the erased interval and suffix to a new token i. Let $(w_{ij}, \mathbf{v}_j)$ denote the attention weight and value vector induced by the original cache $\mathbf{KV}(\mathbf{x})$, let $(\tilde{w}_{ij}, \tilde{\mathbf{v}}_j)$ denote those induced by the exact edited cache $\mathbf{KV}(\tilde{\mathbf{x}})$, and let $(\hat{w}_{ij}, \hat{\mathbf{v}}_j)$ denote those induced by the surrogate cache $\widehat{\mathbf{KV}}(\mathbf{x}; m, n)$. A desired effect is

$$\sum_{m\le j\le n} \hat{w}_{ij}\hat{\mathbf{v}}_j \approx \sum_{n<j\le T} \left(\tilde{w}_{ij}\tilde{\mathbf{v}}_j - w_{ij}\mathbf{v}_j\right). \tag{5}$$

Information Source. The erased span $\mathbf{e}$ and the preserved prefix $\mathbf{p}$ provide the most direct information: the KV states of $\mathbf{e}$ are computed under $\mathbf{p}$, and the downstream contamination carried by the suffix arises because suffix tokens were encoded under a prefix containing both $\mathbf{p}$ and $\mathbf{e}$. It is therefore natural to condition the steering block on the preserved prefix cache together with the erased span.

Parameterization. We generate $\mathbf{KV}^{\text{steer}}(\mathbf{x}; m, n)$ via an eraser E_ϕ. It is a trainable copy of the generator p_θ's backbone, excluding the final language model head. Conditioned on the prefix cache $\mathbf{KV}_{1:m-1}(\mathbf{x})$, it processes the erased span $\mathbf{e}$ to output key and value tensors for positions $m, \cdots, n$. Fig. 1 (a) illustrates the inference pipeline.

Training Objective. During training, we obtain the target continuation from the clean edited prompt $\tilde{\mathbf{x}}$, and optimize the eraser so that the frozen generator produces the same continuation when initialized with the surrogate cache $\widehat{\mathbf{KV}}_\phi(\mathbf{x}; m, n)$. Let $\mathbf{u}$ denote the user query, and let $\mathbf{a} = (a_1, \dots, a_N)$ be the continuation generated from $\tilde{\mathbf{x}}$. We train the eraser with the teacher-forced objective:

$$\mathcal{L}_{\text{erase}}(\phi) = -\sum_{t=1}^{N} \log p_\theta\Big(a_t \mid \widehat{\mathbf{KV}}_\phi(\mathbf{x}; m, n), \mathbf{u}, \mathbf{a}_{<t}\Big). \tag{6}$$

Fig. 1 (b) illustrates the pipeline. The generator parameters θ are frozen and only ϕ is updated.

3.2. Inference-Time Complexity

Recall that $\mathbf{x} = \mathbf{p} \oplus \mathbf{e} \oplus \mathbf{s}$ and $\tilde{\mathbf{x}} = \mathbf{p} \oplus \mathbf{s}$. Constructing KV states for L new tokens after an already cached preserved prefix of length $|\mathbf{p}|$ requires each new token to attend to the prefix and previous new tokens, giving attention cost

$$\sum_{i=1}^{L} (|\mathbf{p}| + i) = O(L(|\mathbf{p}| + L)) \tag{7}$$

Exact erasing reuses the prefix cache $KV_{1:m-1}(\mathbf{x})$ and reruns prefill on the suffix $\mathbf{s}$ under the edited prompt $\tilde{\mathbf{x}} = \mathbf{p} \oplus \mathbf{s}$. Its cache-construction cost is $O(|\mathbf{s}|(|\mathbf{p}|+|\mathbf{s}|))$. When a short span is invalidated early in a long processed context, exact erasing must still rebuild the entire suffix. KVEraser instead constructs the replacement block for the erased interval and reuses the suffix cache. In forward pass, each position in the replacement block attends to the prefix cache and previous positions within the erased interval, so its cache-construction cost is $O(|\mathbf{e}|(|\mathbf{p}| + |\mathbf{e}|))$. KVEraser therefore replaces suffix-length dependence with erased-span-length dependence. In practice, the span to erase, such as a misleading retrieved passage or an incorrect tool result, is often short relative to the suffix, i.e., $|\mathbf{e}| \ll |\mathbf{s}|$. This gives KVEraser a latency advantage over exact recomputation.

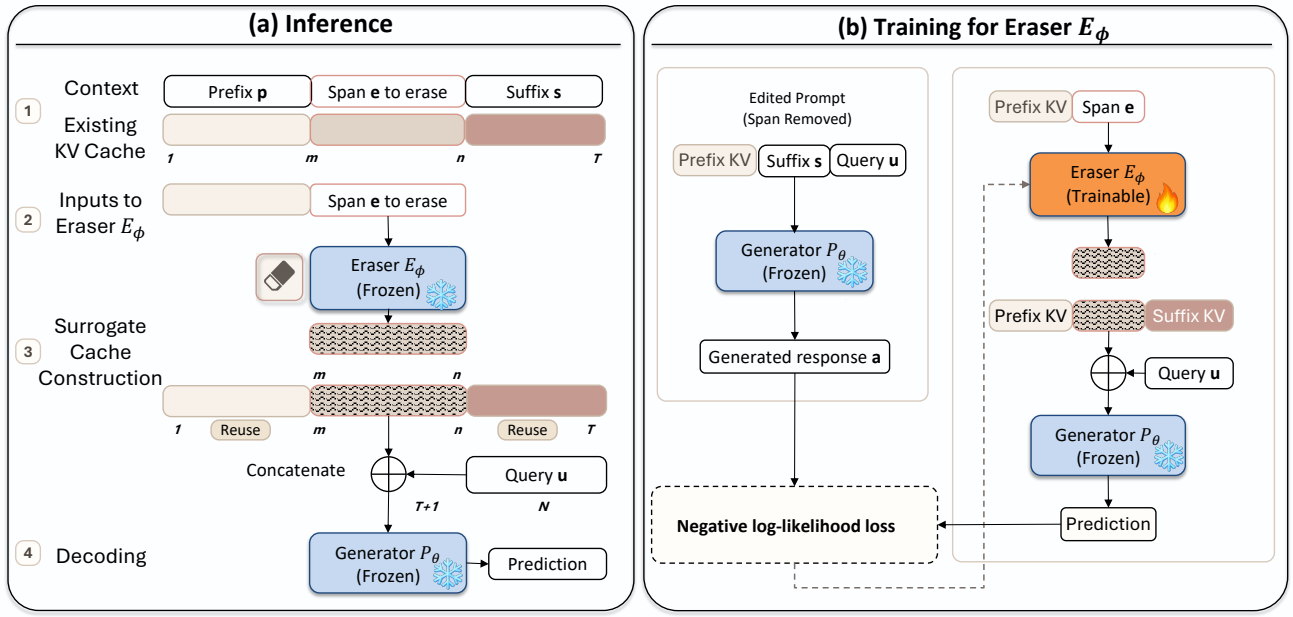

Figure 1. Inference and training pipeline of KVEraser.

4. Experiments

We adopt a two-stage training strategy designed to learn a transferable erasing mechanism.

4.1. Stage 1: Pre-training with Span-Neighbor Retrieval

To construct a sample, we randomly insert a 100-token Wikipedia chunk (span to erase **e**) into a long Wikipedia document. We then select an anchor string either immediately before or after **e**, or elsewhere in the retained context, and ask the model to retrieve the text immediately before or after that anchor. When the anchor is adjacent to **e**, the target neighbor lies across the deleted interval, so successful erasing requires the model to ignore the removed span and recover the surviving neighbor. When the anchor is sampled elsewhere in the retained context, the task instead requires the model to preserve access to non-erased information. To ensure reliable supervision, we retain only samples for which the frozen generator correctly retrieves the target neighboring span under the corresponding clean prompt. We construct 80K pre-training samples from Wikipedia and pre-train the eraser with the teacher-forced objective in Equation 6. See Appendix A for additional details.

4.2. Stage 2: Erasing-Based Task-Specific Fine-Tuning

The 2nd stage adapts the capability to 2 downstream tasks: a controlled synthetic benchmark.

Erasing Needle in A Haystack (NIAH). We propose a controlled benchmark to isolate post-hoc context erasing from parametric knowledge and to stress-test it under long-context retrieval. The benchmark is inspired by multi-value needle-in-a-haystack (NIAH) evaluations for long-context models (Hsieh et al., 2024), but changes the objective from retrieving all inserted facts to selectively removing one of them after the context has already been prefilled.

Each example is constructed by inserting two needles into irrelevant background text to reach a target context size. A needle has the form "One of the special magic numbers for `key` is: `value`", where the key is a random string and the value is a random number. The two needles share the same key but use different values. We insert them at independently sampled, distinct positions in the background text. The earlier needle is designated as the target to erase, while the later needle is retained. We consider context sizes in $\{1\text{K}, 2\text{K}, 4\text{K}, 8\text{K}, 16\text{K}, 32\text{K}\}$ and use 200 training samples for each context size.

Erasing Distractors in QA. Inspired by RAG (Lewis et al., 2020), we consider a natural long document QA setting in which the span to erase is a misleading factual chunk embedded in long documents. We build training samples from three datasets: Natural Questions, TriviaQA, and HotpotQA (Petroni et al., 2021; Kwiatkowski et al., 2019; Joshi et al., 2017; Yang et al., 2018).

We start from questions that the model answers correctly with the original document context. We then randomly insert a 100-token text chunk retrieved from Wikipedia and retain only samples where the added distractor causes the model to answer incorrectly. This filtering ensures that the inserted chunk is harmful rather than merely irrelevant. For example, a query may ask for the name of a star who played

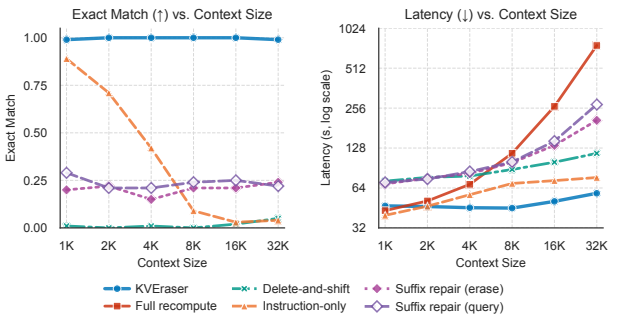

Figure 2. **Erasing a Needle.**

a particular role in a movie. A distractor may discuss another star playing a relevant role in the same movie, causing the model to give an incorrect answer. The result training samples span long contexts, with a median context size of $3.5K$ tokens and a maximum size of $32K$ tokens.

4.3. Setup

We employ Qwen3-8B, which supports a 32K-token context (Yang et al., 2025a). We compare against four baselines.

1. **Full recompute** reuses the valid prefix cache $\mathbf{KV}_{1:m-1}(\mathbf{x})$ and reruns prefill on the suffix $\mathbf{s}$ under the edited prompt $\tilde{\mathbf{x}} = \mathbf{p} \oplus \mathbf{s}$.

2. **Delete-and-shift** removes the cached states of the span and shifts the suffix cache left by $|\mathbf{e}|$ positions.

3. **Instruction-only forgetting** instructs the model to ignore the erased span when answering the query. It relies entirely on LLM's instruction following and reasoning capabilities. Appendix C presents the prompt template.

4. **Local suffix repair** partially recomputes a small contiguous window of suffix tokens while reusing the rest cache. We evaluate two variants that recompute 15% of the suffix: **(i)** one immediately after the erased span, and **(ii)** one near the end of the cache, which is closest to the downstream query. Prior work on long-context inference suggests that attention behavior is strongly position-dependent, motivating these targeted repair locations (Xiao et al., 2024a; Yang et al., 2025b).

4.4. Scaling Behavior for In-Domain Context Erasing

We study the scaling behavior of different approaches using a held-out NIAH subset disjoint from the training samples, with 100 samples for each context size. Since needles are inserted at random positions, larger context sizes lead to larger average suffix lengths. Because the needles are randomly generated, the model cannot rely on parametric knowledge to produce the correct answer without actually erasing the influence of the earlier needle.

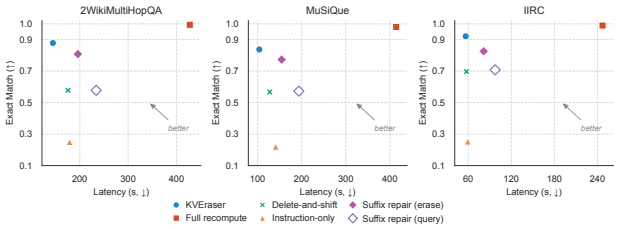

Figure 3. **Erasing Factual Distractors in QA.**

Performance. We evaluate performance with exact match. Fig. 2 shows that KVEraser achieves near-perfect exact match, matching full recompute. Although KVEraser replaces only the KV states of the erased span and keeps the suffix cache, it effectively steers downstream attention and decoding, inducing the same behavior as exact recompute on the edited prompt. The approximate baselines do not provide a reliable alternative. Their post-erasure performance either starts poor or worsens quickly as context grows.

Efficiency. We report latency that excludes the initial prefill of the original context, shared by all methods, and includes all subsequent computation: cache editing or recomputation, query processing, and full decoding. With local cache editing, KVEraser is highly scalable and efficient as context size increases. From 1K to 32K, the latency of KVEraser increases by only 24%, while that of full recompute increases by 17.6×. The approximate baselines' latency grow substantially with context size, and local suffix repair can even be slower than full recompute at smaller context sizes: when erasure fails, the model often generates both needle values, increasing the full decoding time included in measurement.

4.5. Erasing Factual Distractors for Unseen QA Datasets

Following the same procedure as in Sec. 4.2, we prepare evaluation sets from three QA datasets unseen during training: 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and IIRC (Ferguson et al., 2020). Fig. 3 shows that KVEraser remains effective and efficient. Across all datasets, it achieves the highest exact match among approximate baselines while maintaining comparable or lower latency, placing it on the quality–efficiency Pareto frontier. The strongest approximate baseline, local suffix repair (erase), is worse than KVEraser in both exact match and latency. Full recompute attains the highest exact match overall, but its latency is 3–4× that of KVEraser.

5. Conclusion and future work

We introduce KVEraser for context erasing. It achieves near-perfect post-erasure performance on the controlled benchmark across 1K–32K contexts, matching full recomputation, and attains the best quality–efficiency tradeoff among approximate methods on long-document QA.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ahn, J., Seong, I., Kedia, A., Kim, J., Jang, H., Lee, K., and Jeon, Y. LookaheadKV: Fast and Accurate KV Cache Eviction by Glimpsing into the Future without Generation. In *International Conference on Learning Representations*, 2026.

Cai, Z., Zhang, Y., Gao, B., Liu, Y., Liu, T., Lu, K., Xiong, W., Dong, Y., Chang, B., Hu, J., and Wen, X. PyramidKV: Dynamic KV Cache Compression based on Pyramidal Information Funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Chen, Y., Wang, G., Shang, J., Cui, S., Zhang, Z., Liu, T., Wang, S., Sun, Y., Yu, D., and Wu, H. NACL: A General and Effective KV Cache Eviction Framework for LLM at Inference Time. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7913–7926, 2024.

Ferguson, J., Gardner, M., Hajishirzi, H., Khot, T., and Dasigi, P. IIRC: A Dataset of Incomplete Information Reading Comprehension Questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1137–1147, 2020.

Gim, I., Chen, G., Lee, S.-s., Sarda, N., Khandelwal, A., and Zhong, L. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine learning and systems*, 2024.

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., and Fritz, M. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pp. 79–90, 2023.

Ho, X., Duong Nguyen, A.-K., Sugawara, S., and Aizawa, A. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., and Ginsburg, B. RULER: What's the Real Context Size of Your Long-Context Language Models? In *Conference on Language Modeling*, 2024.

Hu, J., Huang, W., Wang, W., Wang, H., Hu, T., Qin, Z., Feng, H., Chen, X., Shan, Y., and Xie, T. EPIC: Efficient Position-Independent Caching for Serving Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 24391–24402, 2025.

Jin, C., Zhang, Z., Jiang, X., Liu, F., Liu, S., Liu, X., and Jin, X. RAGCache: Efficient Knowledge Caching for Retrieval-Augmented Generation. *ACM Trans. Comput. Syst.*, 44(1), 2025.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

Kim, J.-H., Kim, J., Kwon, S., Lee, J. W., Yun, S., and Song, H. O. KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction. In *Advances in Neural Information Processing Systems*, 2026.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, pp. 9459–9474, 2020.

Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. SnapKV: LLM Knows What You are Looking for Before Generation. In *Advances in Neural Information Processing Systems*, pp. 22947–22970, 2024.

Loshchilov, I. and Hutter, F. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

Petroni, F., Piktus, A., Fan, A., Lewis, P., Yazdani, M., De Cao, N., Thorne, J., Jernite, Y., Karpukhin, V., Maillard, J., Plachouras, V., Rocktäschel, T., and Riedel, S. KILT: a Benchmark for Knowledge Intensive Language Tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2523–2544, 2021.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Robertson, S. and Zaragoza, H. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009.

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M. Okapi at TREC-3. In *TREC*, volume 500-225 of *NIST Special Publication*, pp. 109–126, 1994.

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023.

Schmotz, D., Abdelnabi, S., and Andriushchenko, M. Agent Skills Enable a New Class of Realistic and Trivially Simple Prompt Injections. *arXiv preprint arXiv:2510.26328*, 2025.

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E. H., Schärli, N., and Zhou, D. Large Language Models Can Be Easily Distracted by Irrelevant Context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 31210–31227, 2023.

Sun, J., Min, S. Y., Chang, Y., and Bisk, Y. Tools Fail: Detecting Silent Errors in Faulty Tools. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 14272–14289, 2024.

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*, 2017.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient Streaming Language Models with Attention Sinks. In *International Conference on Learning Representations*, 2024a.

Xiao, S., Liu, Z., Zhang, P., Muennighoff, N., Lian, D., and Nie, J.-Y. C-Pack: Packed Resources For General Chinese Embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 641–649, 2024b.

Xie, J., Zhang, K., Chen, J., Lou, R., and Su, Y. Adaptive Chameleon or Stubborn Sloth: Revealing the Behavior of Large Language Models in Knowledge Conflicts. In *International Conference on Learning Representations*, 2024.

Yang, A. et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yang, J., Hou, B., Wei, W., Bao, Y., and Chang, S. KVLink: Accelerating Large Language Models via Efficient KV Cache Reuse. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

Yang, P., Fang, H., and Lin, J. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1253–1256, 2017.

Yang, X., Chen, T., and Chen, B. APE: Faster and Longer Context-Augmented Generation via Adaptive Parallel Encoding. In *International Conference on Learning Representations*, 2025b.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

Yao, J., Li, H., Liu, Y., Ray, S., Cheng, Y., Zhang, Q., Du, K., Lu, S., and Jiang, J. 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.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z. A., and Chen, B. H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Advances in Neural Information Processing Systems*, pp. 34661–34710, 2023.

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C., and Sheng, Y. SGLang: Efficient Execution of Structured Language Model Programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583, 2024.

A. Additional details for pre-training

Data construction for NIAH. Same as (Hsieh et al., 2024), we use sampled Paul Graham's essays for irrelevant background text.

Data construction for QA. We use the 2019-08-01 Wikipedia snapshot. Inspired by the success of hard negative mining in training dense retrievers (Karpukhin et al., 2020), we construct spans to erase via a hybrid retrieval strategy. Specifically, using the first sentence of a long Wikipedia document as the query, we first retrieve candidate Wikipedia documents with BM25 using Anserini (Robertson et al., 1994; Robertson & Zaragoza, 2009; Yang et al., 2017). We then split top-ranked documents into non-overlapping 100-token spans and rerank them with the bge-small-en-v1.5 dense retriever (Xiao et al., 2024b). The prompt template for sample construction is presented below.

Input prompt for pre-training with span-neighbor retrieval

Read and remember the following context for a later question.

{context}

Copy the exact text immediately {before | after} the span <SPAN>{span_text}</SPAN> in the previous context. The copied text may be normal words or a markup tag such as <TEXT>. Reply with the copied text only.

Training. We use a batch size of 8, AdamW (Loshchilov & Hutter, 2019) for optimization, a learning rate of 0.00001, a weight decay of 0.01, and gradient clipping at 1. We train the eraser for 1 epoch.

B. Additional dataset details for task-specific fine-tuning

Data construction. Starting from source samples that the model answers correctly under the clean gold context, we mine harmful distractors from non-gold Wikipedia pages following the same hybrid retrieval strategy described in Appendix A. We keep a candidate span only if, after insertion, the corrupted prompt yields neither an exact match nor any token overlap with the labeled answers (token F1 $= 0$). This filtering ensures that the retained distractors are genuinely misleading rather than merely irrelevant.

Table 1. Size of dataset.

Dataset	Sample size
Natural Questions	1,793
TriviaQA	2,010
HotpotQA	2,488

Table 2. Full context size measured with Qwen3-8B tokenizer.

Dataset	25th percentile	Median	75th percentile	Max
Natural Questions	790	1,880	4,419	25,837
TriviaQA	3,210	7,496	13,711	32,236
HotpotQA	2,046	3,200	5,364	16,858

Table 1 reports the dataset size. Table 2 presents dataset context size statistics.

Training. We use a batch size of 1, AdamW (Loshchilov & Hutter, 2019) for optimization, a learning rate of 0.00001, a weight decay of 0.01, and gradient clipping at 1. We train the eraser for 1 epoch.

Prompt templates. Below we present the prompt templates for NIAH and QA.

Input prompt for NIAH

Extract the requested number(s) from the text.

{context}

Find number(s) in sentence(s) of the form: "One of the special magic numbers for query is: <NUMBER>." Output the <NUMBER>(s) in order as digits only, comma-separated, no spaces, no other text.

Input prompt for QA

Read and remember the following context for a later question.

{context}

Answer the question using only the previously provided context. Respond with a short answer only.

Question: {question}

C. Instruction-only forgetting

Instruction-only forgetting prompt for NIAH

{previous context}

The following previously seen sentence has been deleted and must be ignored when answering: "{deleted_sentence}".

Find number(s) in sentence(s) of the form: "One of the special magic numbers for query is: <NUMBER>." Output the <NUMBER>(s) in order as digits only, comma-separated, no spaces, no other text.

Instruction-only forgetting prompt for QA

{previous context}

The following previously seen passage is distracting and must be ignored when answering: "{distractor_text}"

Answer the question using only the previously provided context. Respond with a short answer only.

Question: {question}

D. Additional dataset details for QA evaluation

Table 3 reports the dataset size for QA evaluation. Table 4 presents dataset context size statistics.

Table 3. Size of evaluation dataset.

Dataset	Sample size
2WikiMultiHopQA	344
MuSiQue	300
IIRC	178

E. Breakdown of failure cases over error categories for QA evaluation

See Table 5.

F. Additional details for ablation studies on information sources for KVEraser

Conditioned on the preserved prefix cache, we use the prompt below to pass query in addition to erased span to KVEraser.

Table 4. Full context size for evaluation datasets, measured with Qwen3-8B tokenizer.

Dataset	25th percentile	Median	75th percentile	Max
2WikiMultiHopQA	3,754	5,066	8,159	15,674
MuSiQue	3,888	5,828	9,814	15,709
IIRC	4,560	6,713	9,702	15,750

Table 5. Breakdown of failure cases over error categories for QA evaluation.

Approach	# Failures	Corrupted-context reliance (%)	Partial overlap (%)	Other (%)
Delete-and-shift	329	51.1%	24%	24.9%
Instruction-only forgetting	625	62.1%	16.5%	21.4%
Local suffix repair (erase)	165	36.4%	32.1%	31.5%
Local suffix repair (query)	325	51.7%	23.1%	25.2%
KVEraser	105	49.5%	21%	29.5%

Input prompt for query conditioning

<KVERASER_QUERY>

{query}

</KVERASER_QUERY>

<KVERASER_STEERING_SPAN>

{erased_span}

G. Ablation study on information sources for KVEraser

We study three alternative conditioning variants for KVEraser. (1) **No prefix**: we drop the conditioning on prefix KV cache $\mathbf{KV}_{1:m-1}(\mathbf{x})$ and pass only the erased span $\mathbf{e}$ to the eraser. (2) **Query conditioned**: we insert the query text $\mathbf{u}$ right before the span $\mathbf{e}$ and use a special marker to separate them. (3) **Suffix conditioned**: motivated by local suffix repair (erase), the strongest approximate baseline in QA, we additionally condition the eraser on 15% of suffix KV following the erased span.

Table 6 shows that our adopted combination of preserved prefix KV and the erased span achieves the best average exact match. Removing the prefix or adding query or suffix information does not yield consistent gains. These results show that our simple, query-agnostic design of KVEraser already captures the most useful information for local cache editing. See Appendix F for more details.

H. Related Work

KV Cache Reuse. Prior work accelerates LLM serving by reusing KV states across requests. Prompt Cache (Gim et al., 2024) and RAGCache (Jin et al., 2025) cache repeated prompt modules or retrieved knowledge across requests when those segments recur in reusable forms. More recent methods such as CacheBlend (Yao et al., 2025), EPIC (Hu et al., 2025), and KVLink (Yang et al., 2026) push reuse beyond identical-prefix settings by recomposing independently cached chunks and repairing the resulting mismatch through selective recomputation, position-independent linking, or trainable cross-chunk tokens.

KV Cache Eviction. Another line of work studies KV cache eviction or compression under a fixed budget. H2O retains heavy-hitter and recent tokens based on attention statistics (Zhang et al., 2023). StreamingLLM stabilizes streaming inference through attention sinks (Xiao et al., 2024a). Later methods further introduce more advanced techniques for cache selection (Li et al., 2024; Cai et al., 2024; Chen et al., 2024; Ahn et al., 2026; Kim et al., 2026). The objective throughout this line of studies is to discard the least useful states while preserving generation quality.

Table 6. Exact match (↑) of KVEraser variants. Best bolded; second-best underlined.

Variant	Prefix KV	Erased span	Query	Suffix KV (15%)	2WikiMultiHopQA	MuSiQue	IIRC	Average
KVEraser	✓	✓			<u>0.878</u>	<u>0.837</u>	**0.921**	**0.879**
No prefix		✓			0.866	**0.850**	0.860	0.859
Query conditioned	✓	✓	✓		**0.892**	0.827	<u>0.893</u>	<u>0.871</u>
Suffix conditioned	✓	✓		✓	0.840	<u>0.837</u>	0.854	0.844

Both cache reuse and cache eviction rely on assumptions that do not hold in our setting. Cache reuse assumes that the reused chunks remain valid parts of the target prompt, while cache eviction assumes that the removed states are unimportant or redundant. In **context erasing**, the designated span is invalid but may still be **influential and informative** for future queries if left in place. Moreover, its effect is not confined to its own KV entries, because later suffix states were computed under a prefix containing that span. Consequently, neither chunk reuse nor KV eviction can recover the counterfactual behavior required for span erasing.

I. Compute resource disclosure

We use 2 80G A100 GPUs for training, and 1 80G A100 GPU for inference. The GPUs are available via a cloud provider. The instance we use has 1.7 TiB RAM and 24 CPU cores. Pre-training takes less than a day. Fine-tuning takes about 8 hours.