KVPRIOR: ON THE SUPRISING EFFECTIVENESS OF MARGINALIZING KVCACHES OVER PREFIXES

Anonymous authors
Paper under double-blind review

ABSTRACT

To avoid costly prefills, non-prefix KVCache methods try to reuse individually prefilled KV caches in non-prefix positions. To recover the accuracy degradation, existing methods either perform inference-time recomputation or amortize its cost with model fine-tuning or link-token training, with little theoretical explanation for why they work. We propose KVPrior, a surprisingly simple *recomputation-free*, *training-free* method to obtain prefix-independent KVCache. Assuming the independence between retrieved documents, it samples a random set of prefixes from the training set, prefill the caches, and averages the context-dependent noise out. This surprisingly simple method nearly recovers the full prefill accuracy and surpassed existing approaches in tasks where the document independence assumptions hold (Biography and Needle-in-the-Haystack). In multi-hop-reasoning datasets where documents depend on each other, we observed weaker yet non-negligible improvements, and combination with existing approaches achieved the state-of-the-art performance.

1 INTRODUCTION

Retrieval-Augmented Generation (RAG) has become a standard approach for extending large language models (LLMs) with external knowledge. By retrieving relevant documents from a large corpus and conditioning generation on those documents, RAG enables language models to answer knowledge-intensive queries without requiring all information to be memorized in the model parameters. As the context length of modern transformer models continues to increase, however, the computational cost of processing retrieved documents has become a major bottleneck. In particular, the prefill stage, which computes the key-value (KV) cache for all input tokens before autoregressive decoding begins, scales linearly with the total context length and often dominates inference latency.

A natural observation is that retrieved documents are frequently reused across many user queries. This has motivated a growing body of work on KV cache reuse, where documents are individually prefilled offline with an empy prefix and their KV caches are stored for later retrieval. Ideally, a serving system could simply concatenate the precomputed KV caches of the retrieved documents together with the query, thereby eliminating the expensive prefill computation almost entirely. Unfortunately, transformer KV caches are inherently prefix-dependent: the hidden representations produced for a document depend on all preceding tokens in the sequence. As a result, KV caches generated independently cannot be naively concatenated without introducing substantial accuracy degradation.

Recent work has proposed several approaches to overcome this limitation. Existing methods mitigate prefix dependence by partially recomputing the cache online Yao et al. (2025); Hu et al. (2025); Yan et al. (2026); Wang et al. (2026), fine-tuning the language model to become more tolerant to cache composition Yang et al. (2026), or learning auxiliary link-token embeddings that bridge independently prefilled documents Chen et al. (2026). While these methods substantially improve performance, they incur additional costs during deployment, require model modification or extra training, or rely on a mechanism that remains poorly understood (What are these training methods actually learning?)

In this work, we revisit the source of performance degradations in non-prefix cache reuse. Building upon the observations made by KVPacket, we hypothesize that the degradation arises from two distinct factors: (A) The loss of cross-attention between documents, and (B) position-specific/empty-prefix-specfic artifacts in the cached KV, often observed as "attention sink" Hu et al. (2025); Chen et al. (2026), that persist even after correcting for Rotary Potisional Embedding Su et al. (2024) and distort

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.

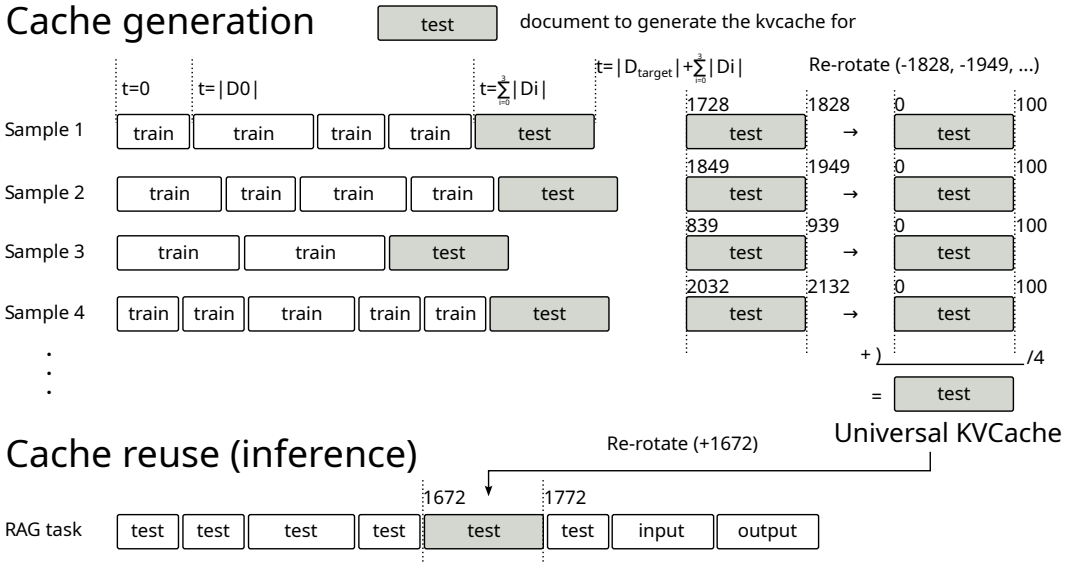


Figure 1: KVPrior constructs a position/prefix-independent KV cache by marginalizing the cache over a set of randomly sampled irrelevant prefix documents.

the attention score distribution. We hypothesized that the source A fundamentally requires partial cache recomputations with the cross-document context while a completely offline method would only address the source B. A performant offline method however can address B and localize the degradation to A, thus offers a separation of concerns and opens possibility to orthogonal improvements.

This perspective suggests a surprisingly simple solution by focusing our attention to removing the artifacts. We propose KVPrior (Fig. 1): a cache representation that is invariant to *the position and the prefix*. In many RAG applications, retrieved documents mostly (though not necessarily) contribute independently to answering the query, and the specific prefix preceding each document is largely incidental. Inspired by Bayesian statistics, we intend to turn this intuition into a prior assumption. Under the independence assumption as a prior, any contribution from the prefix, including the empty one, is a noise/bias. To remove the bias, we average it out over randomly sampled prefixes drawn from the training corpus or from an unrelated dataset that has no chance of data leakage. Offline, each document is prefilled multiple times under different random prefixes, and the resulting KV caches are averaged to suppress context-dependent artifacts while preserving document-specific information. The resulting prefix-independent cache can then be reused directly at inference time without requiring any online recomputation or model modification when the prior assumption holds. When the prior assumption does not hold, existing recomputation-based non-prefix cache reuse methods can be seen as a posterior update which further improves the accuracy.

Despite its simplicity, KVPrior nearly recovers the accuracy of full prefill while remaining entirely offline and training-free. *The number of samples needed is as few as 4* to achieve substantial improvements, though more samples help. Unlike previous approaches, it requires neither model fine-tuning nor the optimization of auxiliary embeddings, thus it requires minimal computational resources, and it introduces no additional runtime overhead if used alone. On retrieval tasks where retrieved documents are largely independent, including Biography and RULER (Needle-in-the-Haystack) , KVPrior matches or surpasses existing methods. On multi-hop reasoning benchmarks, where interactions between retrieved documents are essential and cross-attention is necessary, KVPrior alone provides more modest, but still substantial improvements, and when combined with cache recomputation methods, it further improves the accuracy.

Our contributions are as follows:

- We provide a conceptual decomposition of prefix dependence into two sources of error: missing cross-attention and context-dependent artifacts in the KV cache.

- We introduce KVPrior, a simple offline procedure that removes context-dependent noise by averaging KV caches over randomly sampled prefixes.
- KVPrior nearly recovers full-prefill performance on RAG tasks with independent documents.
- KVPrior is complementary to existing non-prefix KV cache methods and can be combined with them to achieve state-of-the-art results on challenging multi-hop reasoning benchmarks.

2 PRELIMINARY

Let $1..N$ denote a list of indices $(1, \cdots, N)$. *Auto-regressive generative model* is a probability distribution $p(\mathbf{c}) = p(\mathrm{c}_1, \cdots, \mathrm{c}_N)$ where the random variable $\mathbf{c}$ represents a vector of tokens, typically associated with a word or a sub-word token in natural language. Transformer-based language models (LMs) Vaswani et al. (2017) decompose it into each conditional distribution modeled by a large set of parameters θ: $p(\mathbf{c}) = p_\theta(\mathbf{c}_N|\mathbf{c}_{1..N-1})p(\mathbf{c}_{1..N-1}) = \cdots = \prod_{i=j}^{N-1} p_\theta(\mathbf{c}_i|\mathbf{c}_{1..i-1})p(\mathbf{c}_{1..j})$, where each conditional distribution $p_\theta(\mathbf{c}_i|\mathbf{c}_{1..i-1})$ performs a next-token prediction.

Transformer (Vaswani et al., 2017) architecture performs autoregressive generation using layers of multi-head self-attention mechanism. Given a hidden state vector $\boldsymbol{h}[l]$ at layer l, a single self-attention with Rotary Potisional Embedding (Su et al., 2024, RoPE) is as follows:

$$\boldsymbol{x} = \mathrm{Norm}(\boldsymbol{h}[l]), \quad \begin{array}{ll} K = & R(\boldsymbol{t})(W^K\boldsymbol{x}), \\ Q = & R(\boldsymbol{t})(W^Q\boldsymbol{x}), \\ V = & W^V\boldsymbol{x}, \end{array} \quad O = V \odot \textsc{softmax}\left(\frac{Q^\top K}{\sqrt{2d}}\right). \tag{1}$$

$\boldsymbol{t} = 1..N$ is a position and $R(\boldsymbol{t})$ is a RoPE operator. For H self-attentions, the outputs $O_1, \cdots, O_H$ are concatenated and fed to an MLP to produce $\boldsymbol{h}[l+1] = \mathrm{MLP}(O_1; \cdots; O_H)$.

Given a $2d$-dimensional embedding $\boldsymbol{x}$, RoPE rotates $\boldsymbol{x}$ to position t as follows. It maintains the following invariance that makes resetting the positions from t to s possible:

$$R(t)\boldsymbol{x} = \begin{bmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{0:d} \\ \boldsymbol{x}_{d:2d} \end{bmatrix}. \qquad \begin{array}{c} R(t+s) = R(t)R(s). \\ \therefore R(s)\boldsymbol{x} = R(s-t)(R(t)\boldsymbol{x}). \end{array}$$

Modern inference engines avoids the $O(N^2)$ complexity in computing $Q^\top K$ by caching the submatrix of K. Given $\boldsymbol{h}[l]$ and K, the submatrix $K_{1..N-1}$ is identical to the output of the previous iteration, thus the $N-1$ token prefix can be cached. The same applies to V.

$$\boldsymbol{h}[l] = \begin{bmatrix} \boldsymbol{h}[l]_{1..N-1} \\ \boldsymbol{h}[l]_{N-1..N} \end{bmatrix}. \quad K = R(\boldsymbol{t})W^K h[l] = \begin{bmatrix} R(\boldsymbol{t})W^K\boldsymbol{h}[l]_{1..N-1} \\ R(\boldsymbol{t})W^K\boldsymbol{h}[l]_{N-1..N} \end{bmatrix} = \begin{bmatrix} K_{1..N-1} \\ K_{N-1..N} \end{bmatrix}.$$

Retrieval Augmented Generation (RAG) task is a set of 4-tuples $\Pi = (Q, A, D, S)$ where S is a set of documents, Q is the question, A is the ground-truth answer, and $D = (D_1, \cdots D_K)$ is a sequence of K gold retrieval documents that contain information relevant to the Q and A. We assume availability of D and focus on generating answers A; Retrieving D from a larger document corpus S is outside the scope of this paper. Using a language model $p_\theta(A|Q, D)$, we sample an answer $\widetilde{A}$ and compare it against A. The system prompt is implicitly added to D as a document D_1 if necessary.

Computing K and V is often called *prefilling*. We refer to the standard autoregressive prefilling with prefix caching as FULL_RECOMPUTE. Prompts that differ in non-prefix positions invalidate the cache, forcing recomputation of K and V after the first token that differs from the cached tokens. To address this, existing work attempts to discover a way to reuse KV caches in non-prefix positions, which is refered to as various names (e.g., position-independent caching, non-prefix KV cache reuse, individual prefill). The naive baseline method, NO_RECOMPUTE, for RAG works as follows. For each D_k, let C_k be the corresponding individually-prefilled KV cache produced by feeding the tokens in D_k by itself. Since C_k has RoPE for positions $1..|D_k|$, it must be re-rotated to the correct position before concatenation during the inference. Let C'_k be the rerotated cache. Then the entire concatenated cache $\widetilde{C}^{\mathrm{NR}}$ is as follows. $\widetilde{C}^{\mathrm{NR}}$ is then used to prefill Q and generate an answer $\widetilde{A}$.

$$\widetilde{C}^{\mathrm{NR}} = C'_1; \cdots; C'_K, \quad C'_k = R(\mathbf{1} \cdot \textstyle\sum_{i=1}^{k-1} |D_i|)(C_k), \quad \mathbf{1} = (1, 1, \cdots, 1).$$

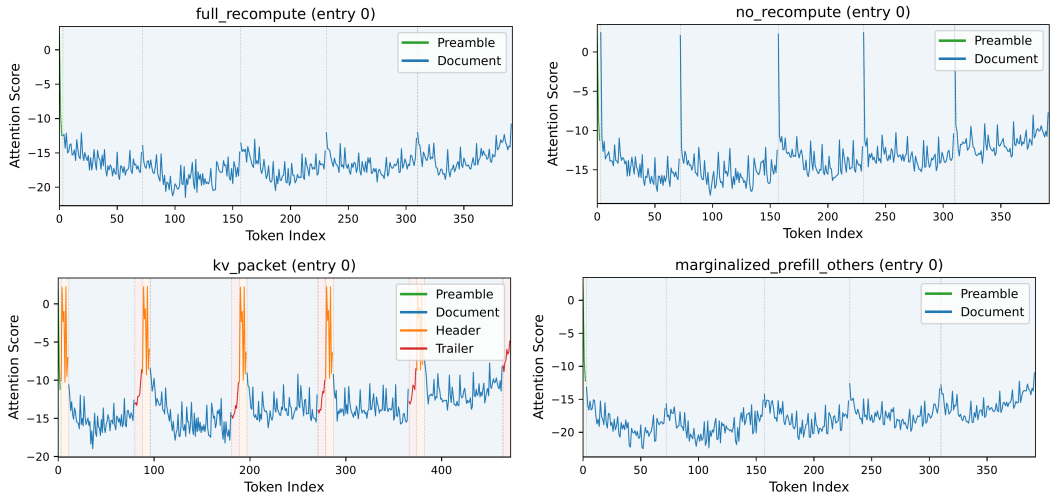


Figure 2: Pre-softmax logits summed over attention heads, layers, and query tokens (y-axis), plotted over document token positions in the x-axis. FULL_RECOMPUTE (top left) shows the scores for the regular prefilling. NO_RECOMPUTE (top right) causes attention spikes at every beginning of the documents, as identified by EPIC Hu et al. (2025). KVPacket's shared learned embeddings Chen et al. (2026) absorbs those attention scores, while minimizing the effect to the semantic meaning. KVPrior's attention score closely follows the scores of FULL_RECOMPUTE via a simple averaging that is performed offline with no training.

NO_RECOMPUTE causes serious accuracy degradations and a number of methods have been addressed. Majority of methods belong to a category called *selective recomputation*, which selects several token positions according to some criteria and recomputes K and V for those positions. However, recomputation must happen after the combination of documents D is revealed, which affects the inference speed. Recomputation methods often allows scaling the number of tokens to recompute, and can theoretically match the FULL_RECOMPUTE accuracy when all tokens are recomputed. In practice, the user finds a trade-off between the latency and the accuracy.

Another group of methods that address the degradation are based on fine-tuning and link token embeddings. KVLink (Yang et al., 2026) pioneered this method, which inserts additional tokens between documents and fine-tunes the model and the embeddings to match the original generation via distillation loss. However, this method also has a runtime penalty because the link tokens must be prefilled online. It also requires a costly infrastructure for a full finetuning when applied to a large model.

KVPacket (Chen et al., 2026) removes these weaknesses by training only the link tokens to reduce the resoure requirement, and prefill link tokens together with each document offline, which completely eliminates the runtime cost.

3 The Symptom: Context-Sensitive Artifacts

Let us reiterate the hypothesis outlined in the introduction.

Hypothesis 1. *The degradation in non-prefix kvcache, in particular* NO_RECOMPUTE*, is caused by*

(A) the loss of cross-attention between documents, and

(B) the artifacts in the cached KV due to the specific potision (individual prefill starts from position 0) and the specific prefix (individual prefill uses an empty prefix).

This paper focuses on addressing (B), while delegating (A) to existing methods that incorpolate some degree of cache recomputation that addresses cross-attentions at the cost of runtime penaly. To our knowledge, the only method that works in this regime is KVPacket (Chen et al., 2026), which is

guaranteed not to address item A. Their link tokens are prefilled together with each document D_k without seeing other documents.

To visualize the artifact, we plotted the attention scores from the Q vectors of the user query to the K vectors of retrieved documents with FULL_RECOMPUTE and NO_RECOMPUTE. We also compare the same scores when we apply KVPacket, reproducing the experiments by Chen et al., although they did not show the scores for FULL_RECOMPUTE.

In Fig. 2, y-axis shows the pre-softmax logits summed over attention heads, layers, and query tokens, plotted over document token positions in the x-axis. NO_RECOMPUTE **(b)** shows that Q vectors detect the beginning of each document from each C'_k even though their positional embedding is rotated. This is also the main motivation of EPIC (Hu et al., 2025), which recomputes the first few tokens of each document exclusively.

KVPacket **(c)** appears to have learned to absorb the attention spike so that the main documents do not incorrecly receive such attention spikes. It does not, however, address the attention spikes itself.

We observe from FULL_RECOMPUTE **(a)** that the 1st document has a similar attention spike (which is not an issue, since it is actually in the beginning of the prompt) while the 2nd and later documents not only lacks the spikes, but has a similar score distribution overall. In other words, just putting some tokens as a prefix alone may suppress the attention spike.

Further, in **(d)** we experimented with a variation of KVPacket which simply uses the average of the entire vocaburary embedings as a link embedding. Surprisingly, this also reduced the attention spike, and as we later see, it even has a minor accuracy improvements.

These results suggest that all that KVPacket is doing is merely hiding the attention spike in a couple of model-specific meaning-free tokens so that an incorrect attention would not affect the later tokens. If we device a new, much simpler method that removes the spike entirely, then there is no reason to train any embeddings at all.

4 THE SOLUTION: KVPRIOR

We propose a surprisingly simple, completely offline, training-free method that addresses the issue B. Our goal is to obtain a KV representation that is free of these position/prefix-dependent artifacts. As seen in Fig. 2, *non-prefix positions tend not to have spikes*. Therefore, one intuitive way to remove the attention spike is to append a random, meaningless/irrelevant prefix to the target document, prefill the KV cache, then remove the prefix. Obviously this leaves some trace of the irrelevant information in the KV cache, which may harm the accuracy downstream. However, also intuitively, those unnecessary information can be removed by sampling many (≈ 8) random prefixes and averaging them out. We formalize this simple intuition in Alg. 1 and Fig. 1.

Algorithm 1 KVPrior: Prefix-independent kvcache generation for document D with N samples and the maximum number of prefix documents M. It assumes a uniform distribution $p_{\text{train}}(D)$ of documents from the train set.

for $i \in 1..N$ **do**
  $m \sim \text{Uniform}(1, M)$
  for $j \in 1..m$ **do**
    $D_{ij} \sim p_{\text{train}}(D)$
  $C_i = (K_i, V_i) \leftarrow \text{prefill}(D_{i0}; \cdots D_{im}; D)$
return $\tilde{C} = \frac{1}{N}\sum_{i=1}^{N} C_i = \left(\frac{\sum K_i}{N}, \frac{\sum V_i}{N}\right)$

We interpret this method from a Bayesian point of view. Our prior assumption is that *each document is independent from each other*, and thus independently contributes to the interpretation of the question and the generation of the answer. As this mean-field assumption is merely a prior assumption, it is *possible and permissible* to be wrong; A prior is meant to be updated via observation into a posterior. This also clarifies the role of recompution-based methods; If they are combined with KVPrior, they can be seceretly interpreted as performing a posterior update to the KV cache.

4.1 THEORETICAL JUSTIFICATION

To formalize KVPrior, we come back to the generative modeling perspective of autoregressive language model. We model KV caches as a random variable that depends on the prefix documents.

Let $\mathbf{z} = \mathbf{c}_{1..N-1}$. Recall that the conditional distribution of token N is modeled as $p(\mathbf{c}_N) = \mathbb{E}_{p(\mathbf{z})}[p(\mathbf{c}_N|\mathbf{z})]$. Let $C(\mathbf{z})$ be the cache of $\mathbf{z} = \mathbf{c}_{1..N-1}$, which is a deterministic function. In the transformer archtecture, $p(\mathbf{c}_N|\mathbf{z})$ is also a deterministic function $p(\mathbf{c}_N|\mathbf{z}) = f(C(\mathbf{z}))$. Therefore

$$p(\mathbf{c}_N) = \mathbb{E}_{p(\mathbf{z})}[p(\mathbf{c}_N|\mathbf{z})] = \mathbb{E}_{p(\mathbf{z})}[f(C(\mathbf{z}))]. \tag{2}$$

Our goal is to approximate this with $\tilde{C} = \frac{1}{M}\sum_{i=1}^{M} C_i$ (Alg. 1, line 6). Let $\mu = \mathbb{E}_{p(\mathbf{z})}[C(\mathbf{z})]$ and $C(\mathbf{z}) = \mu + \epsilon$. Then the Taylor expansion of f around μ is

$$f(C(\mathbf{z})) = f(\mu + \epsilon) = f(\mu) + J_f(\mu)\epsilon + \frac{1}{2}\epsilon^\top H_f(\mu)\epsilon + O(\|\epsilon\|^3). \tag{3}$$

$$\mathbb{E}_{p(\mathbf{z})}[f(C(\mathbf{z}))] = f(\mu) + J_f(\mu)\cdot 0 + \frac{1}{2}\mathbb{E}_{p(\mathbf{z})}[\epsilon^\top H_f(\mu)\epsilon] + O(\mathbb{E}[\|\epsilon\|^3]) \tag{4}$$

$$\approx f(\mu) = f(\mathbb{E}_{p(\mathbf{z})}[C(\mathbf{z})]) \tag{5}$$

$$\approx f(\tilde{C}). \quad \text{(Monte-Carlo estimate.)} \tag{6}$$

The key step in this approximation is Eq. 5 which assumes that the hessian of f around μ in the direction of cache variation ϵ is small. Intuitively, this is interpreted as *since we randomly sample documents* $\mathbf{z}$ *that are irrelevant to* $\mathbf{c}_N$*,* ϵ *is orthogonal to* H_f *and does not affect* f *and* $p(\mathbf{c}_N)$. We empirically validate this claim in Sec. 5.

We also note that this contains another approaximation; The formulation above models a token-wise generative distribution rather than the document-wise distribution, while in the actual KVPrior the tokens within each document depends causally to the earlier tokens. Existing work on speculative decoding and diffusion-based decoding shows that this mean-field assumption over the token positons is also a still relatively accurate model of the original distribution. We skip the discussion of this approximation based on this insight.

5 EMPIRICAL EVALUATION

The performance of generation is measured by two metrics: *F1 metric*, which measures the token-level Jaccard distance between A and $\tilde{A}$, and *TTFT* (Time to First Token), which measures the sum of runtimes for any additional processing for KV caches and the runtime for prefilling the query Q. Due to the different trade-off on the accuracy and the runtime, proper evaluation requires comparing the runtime-accuracy pareto front.

In addition, LLMs are known to exhibit positional biases . To remove the bias, we compute the scores over 6 random permitations of the original document ordering in the dataset, including the single original ordering, and compute the median score and the width (difference between minimum and the maximum). The width shows the sensitivity to the document ordering, which is lower the better.

Our KVPrior code is heavily based on the public repository of KVPacket, which uses the bare torch/Huggingface engine as a backend. Integration of KVPrior with State-of-the-Art inference engines such as vllm (Kwon et al., 2023) and SGLang (Zheng et al., 2024) to improve the throughput for a massive number of simultaneous requests is future work but is clearly outside the scope of this paper.

In all experiments, we evaluated four models: Llama 3.1 8B (), Qwen 3 Instruct 4207 (), Granite 4.1 3B and 8B ().

We used five RAG datasets: Biography (), RULER / Needle-in-a-Haystack (Niah) (), HotpotQA , Musique , TwoWikiQA . Two of the five datasets, Biography and RULER, consist of tasks where documents do not depend on each other and answering the questions do not require combining information from multiple documents, and therefore the document independence / mean-field assumption holds. The rest of the datasets all require multihop reasoning.

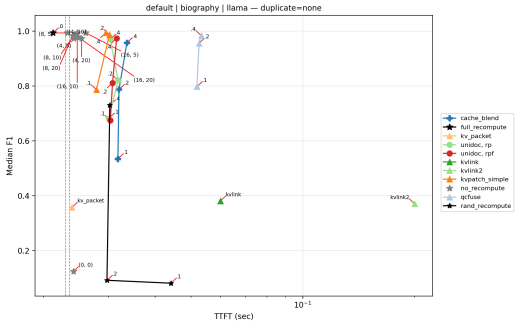


Figure 3:

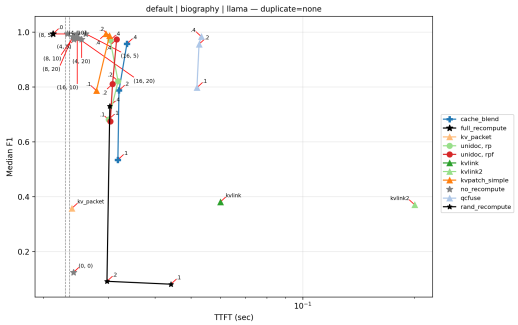


Figure 4:

5.1 Evaluations in Non-Multi-Hop Benchmarks

Benchmarks that contain no multi-hop reasoning and thus no dependency between retrieved documents are ideal targets of KVPrior. We evaluate KVPrior along with other methods on Biography / Ruler.

Fig. 3 shows the accuracy-runtime trade-offs. Since KVPrior is an offline method, it shows no runtime degradation against NO_RECOMPUTE baseline.

In both Biography and Niah, KVPrior nearly approaches the FULL_RECOMPUTE accuracy.

As another zero-recomputation off-line method, KVPacket was outperformed by KVPrior.

Despite their runtime penalty, recomputation-based methods are also outperformed by or are merely comparable to KVPrior.

It is also worth noting that KVPrior shows very little positional bias, as evidenced by the limited magnitude of performance fluctuations compared to others. This is generally preferred, as the user do not have to worry about additional source of non-determinism.

We also plotted several hyperparameters for KVPrior. Generally speaking, more samples tend to improve the performance, but mere 4 samples often provide a significant performance improvement, and we observe saturations at 10 samples.

Another important control experiments is to eliminate the possibility of data leakage; Using the training documents to prefill the prefix of a test document could potentially leak information about the test data if the benchmarks reuse the some set of documents in the training and the test set, even if each RAG task never appears in the training and the test set simultaneously.

To control for this factor, we evalauted a variant that uses random documents from Biography to prefill the documents in Niah, HotpotQA, Musique, TwoWikiQA. Even in this evaluation, KVPrior showed improvements over NO_RECOMPUTE and showed comparable performance as recomputation methods.

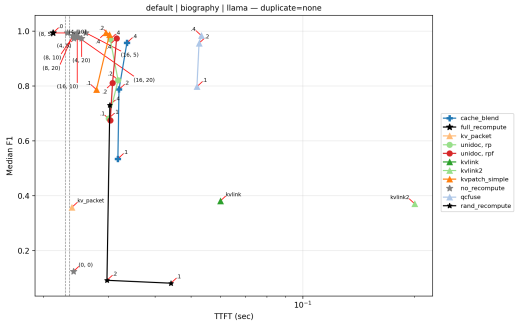


Figure 5:

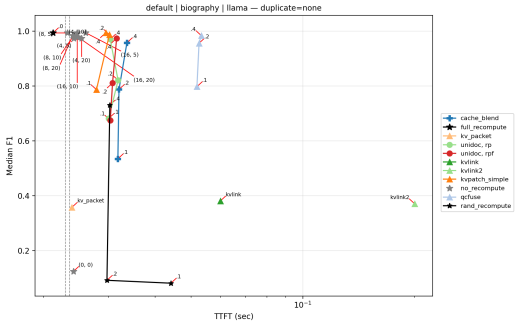


Figure 6:

There is an important observation from this experiment, however: Even if there is no potential for data contamination, the accuracy degraded over the experiments that use the training set of the same benchmark. This implies that the common element shared by the random prefixes, such as the specific document format, remains in the KVCache even after averagng and helps the downstream inference. It poses an interesting hypothesis that the more the random prefixes are closer to the actual prefixes, the better the accuracy, thus a method that samples from a better prefix distribution may further improve the accuracy.

5.2 Evaluations in Multi-Hop Reasoning Benchmarks

In multi-hop benchmarks, KVPrior shows more modest, yet still substantial performance improvements. Fig. 4 compares KVPrior with recompuation methods. KVPrior is on par or slightly less accurate than recomputation-based methods. To improve the accuracy on multi-hop benchmarks, we combine KVPrior with recomputation methods. Specifically, we apply recompuation methods to $\widetilde{C}$ to select some tokens and then recompute sub-tensors of $\widetilde{C}$ in those positions. We see this as a form of posterior updates that further boosts the accuracy.

Fig. 5 shows the performance with $(N, M) = (4, 5)$ and $(8, 10)$. KVPrior+Recompuation generally improves the accuracy over the KVPrior alone, demonstating that such combination is promising.

5.3 Measuring the Hessian Contribution

To validate the theoretical claim in Sec. 4.1, we measure the empirical value of $H(\mu, \epsilon) = \epsilon^\top H_f(\mu)\epsilon$. However, computing the hessian H_f is expensive and infeasible especially for a large model. To avoid it, define g and apply Taylor expansions as follows to obtain its second order approximation.

$$\begin{aligned} g(\mu, \epsilon) &= f(\mu + \epsilon) + f(\mu - \epsilon) - 2f(\mu) \\ &= f(\mu) + J_f(\mu)\epsilon + \frac{1}{2}\epsilon^\top H_f(\mu)\epsilon + O(\|\epsilon\|^3) \end{aligned}$$

	Biography	Niah	HotpotQA	Musique	TwoWikiQA
Llama					
Qwen3					
Granite 3B					
Granite 8B					

Table 1:

$$
\begin{aligned}
&+ f(\mu) + J_f(\mu)(-\epsilon) + \frac{1}{2}(-\epsilon)^\top H_f(\mu)(-\epsilon) + O(\|\epsilon\|^3) \quad - 2f(\mu) \\
&= \epsilon^\top H_f(\mu)\epsilon + O(\|\epsilon\|^3) \approx H(\mu, \epsilon).
\end{aligned}
$$

The value of interest is the relative impact of $\frac{1}{2}\mathbb{E}_{p(\mathbf{z})}[H(\mu, \epsilon)]$ on $f(\mu)$, where $\epsilon = C(\mathbf{z}) - \mu$. The impact can be quantified as a ratio $R = \frac{\frac{1}{2}\mathbb{E}_{p(\mathbf{z})}[H(\mu,\epsilon)]}{f(\mu)}$ on each token in a document. We averaged R over all tokens in each document D_k, over all documents $D = (D_1, \cdots, D_K)$ in each task, over all tasks in the validation dataset of each benchmark. Since the true cache mean μ is not available, we replace it with the empirical mean $\widetilde{C}$.

Table 1 shows the results on four models and five benchmarks. The value tends to be below XXX in Biography and Niah, supporting our theoretical claim of why KVprior works. The value tends to be larger but still below YYY in the multi-hop datasets.

6 Related Work

There are multiple evidences to our claim in the existing work that even after the rerotation of RoPE embedding, there is a chance that position-specific artifacts remain in the KV cache. NoPE showed that positional embedding is a useful prior in the beginning of the training, but in the later phase of training, it is unnecessary. This implies that the model acquired its own mechanism for encoding the position in the hidden space. DroPE further supports this claim, discovering that it is possible to fine-tune a model to work without positional embeddings while retaining the near-identical performacne.

7 Conclusion

We proposed KVPrior, a position-and-prefix-independent KVCache prefilling method that works on a variety of tasks. Inspired by Bayesian statistics, it uses the document independence as a prior assumpiton, and creates a KV cache that tends to work with variety of positions and prefix.

We believe ours is the first characterization of position-independent KV Cache from a Bayesian point of view, which opens a variety of potential future work. For example, while our method mitigates the bias by averaging, it does not fully represent the prior distribution of the KVCache as e.g. a Gaussian. We do not propose any specialized mechanism for updating the prior into a posterior either. For example, an interesting avenue of future work includes a custom Lora adapter that is taylored toward computing the differential update to the KV cache.

References

Masataro Asai and Alex Fukunaga. Tiebreaking Strategies for Classical Planning Using A^* Search. In *Proc. of AAAI Conference on Artificial Intelligence*, Arizona, USA, February 2016.

Chuangtao Chen, Grace Li Zhang, Xunzhao Yin, Cheng Zhuo, Bing Li, and Ulf Schlichtmann. KV Packet: Recomputation-Free Context-Independent KV Caching for LLMs. *arXiv preprint arXiv:2604.13226*, 2026.

Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Tiancheng Hu, Zhang Qin, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. EPIC: Efficient position-independent caching for serving large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp,

Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proc. of the International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24391–24402. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/hu25j.html.

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.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, pp. 5998–6008, 2017.

Shihao Wang, Jiahao Chen, Yanqi Pan, Hao Huang, Yichen Hao, Xiangyu Zou, Wen Xia, Chongyang Qiu, Wentao Zhang, and PengFei Wang. ProphetKV: User-Query-Driven Selective Recomputation for Efficient KV Cache Reuse in Retrieval-Augmented Generation. In *Proc. of the International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=SwGwmKIgUb.

Jianxin Yan, Wangze Ni, Zhenxin Li, Jiabao Jin, Zhitao Shen, Haoyang Li, Jia Zhu, Peng Cheng, Xuemin Lin, Lei Chen, et al. QCFuse: Query-Aware Cache Fusion via Compressed View for Efficient RAG Serving. *arXiv preprint arXiv:2606.05875*, 2026.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. KVlink: Accelerating Large Language Models via Efficient KV Cache Reuse. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 38:133797–133824, 2026.

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 *Proc. of the European Conference on Computer Systems*, pp. 94–109, 2025.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. SGLANG: Efficient Execution of Structured Language Model Programs. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 37:62557–62583, 2024.

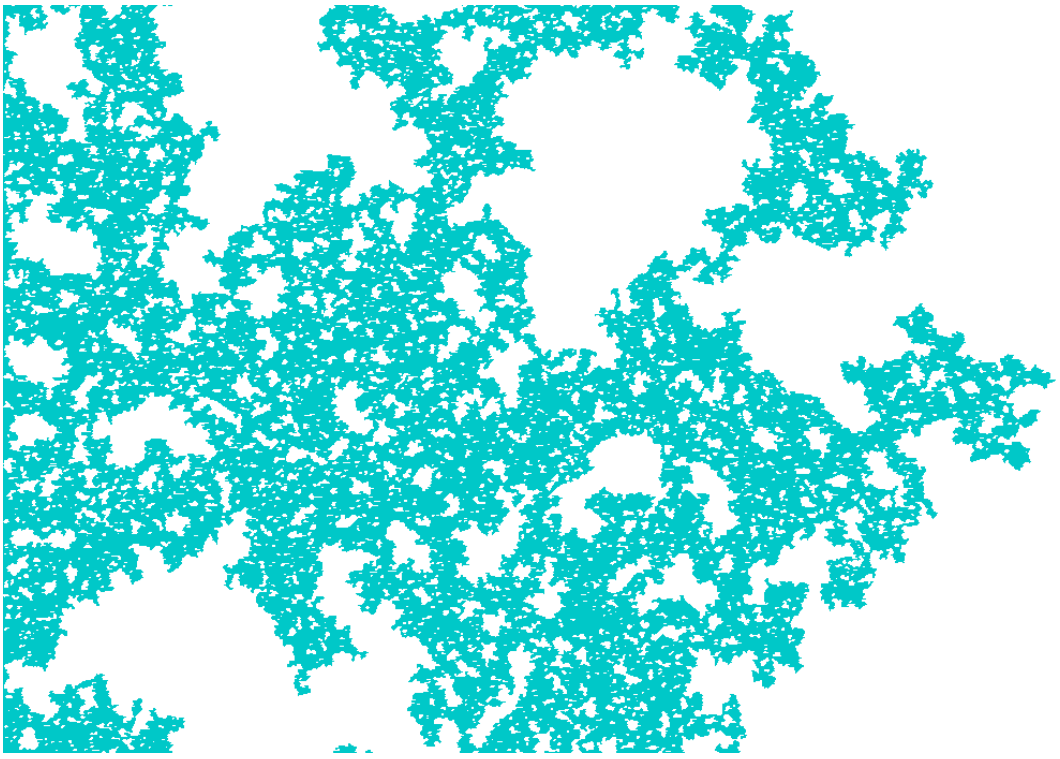


Figure A7: This is Invasion Percolation

APPENDIX

A1 INTRODUCTION

BBB! Fig. A7 Asai & Fukunaga (2016)