On the Efficacy of Eviction Policy for Key-Value Constrained Generative Language Model Inference

Anonymous ACL submission

Abstract

Large language Models (LLMs) are notably cost-prohibitive to deploy in resource-constrained environments due to their excessive memory and computational demands. In addition to model parameters, the key-value cache is also stored in GPU memory, growing linearly with batch size and sequence length. As a remedy, recent works have proposed various eviction policies for maintaining the overhead of key-value cache under a given budget. This paper embarks on the efficacy of existing eviction policies in terms of *importance score calculation* and *eviction scope construction*. We identify the deficiency of prior policies in these two aspects and introduce RoCo, a robust cache omission policy based on local attention scores and robustness measures. Extensive experimentation spanning prefilling and auto-regressive decoding stages validates the superiority of RoCo. Finally, we release EasyKV, a versatile software package dedicated to user-friendly key-value constrained generative inference. Code available at https://anonymous.4open.science/r/EasyKV-9088/.

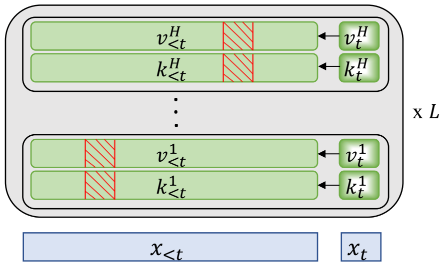

Figure 1: Illustration of KV cache eviction inside one attention layer (L in total). In this example, a single pair of key-value vectors are deleted (red hatched areas) before appending the next token's. Different heads (H in total) at model layers may evict at different positions.

1 Introduction

Recent advancements in Large Language Models (LLMs) have demonstrated outstanding proficiency in a wide range of text generation scenarios, such as content creation, abstractive text summarization, and instruction-following (Thoppilan et al., 2022; Wei et al., 2022; Wang et al., 2023; Touvron et al., 2023a,b). Nevertheless, deploying LLMs is a notably costly undertaking considering their tremendous parameter size and quadratic cost of attention layers. Accordingly, model compression (Frantar and Alistarh, 2023; Xia et al., 2023) and memory-efficient attention (Dao et al., 2022b; Dao, 2023) techniques have emerged to tackle these challenges and achieved substantial outcomes.

Owning to the auto-regressive nature of LLM inference (Vaswani et al., 2017; Radford et al., 2019), the intermediate attention key-value vectors are also required to be stored in memory to avoid redundant key-value projection in future steps. The size of key-value cache (KV cache) depends on the configuration of the attention layer, batch size, and sequence length, which poses challenges in both memory footprint, I/O cost, and computation burden given the increasingly upsoaring model scale and user requests. Variants like multi-query attention and grouped-query attention (Shazeer, 2019; Ainslie et al., 2023) reduce the size of the KV cache with fewer attention heads, but cannot be directly applied to pre-trained LLMs without re-training.

In pursuit of flexible and training-free control over KV cache during LLM inference, recent works (Liu et al., 2023b; Zhang et al., 2023; Xiao et al., 2023b; Oren et al., 2024) have investigated implementing KV cache with *eviction policy*, where the key-value vectors of certain tokens are strategically deleted from memory (see Figure 1). In this way, the size of the KV cache can be maintained under a specified budget, leading to decreased memory and computational overhead. Despite the claimed and empirically observed reduction in KV cache, there still lacks a comprehensive comparative analysis of these methods. This study aims to fill this gap and embarks on the efficacy of existing eviction policies from a unified framework, which decomposes an eviction policy into two design dimensions: *importance score calculation* and *eviction scope construction*. The former characterizes how important a pair of key-value vectors is to future generations, while the latter determines which tokens are readily allowed to be evicted from the cache We categorize existing eviction policies according to these two dimensions. Following our preliminary analysis, we discover that the way current methods calculate importance scores utilizing local statistics can only weakly approximate that derived from global statistics (full KV cache without eviction). Moreover, prior methods commonly construct the eviction scope by only incorporating tokens outside of a local window, which we show endure high sensitivity to window size.

In this paper, we propose RoCo, a Robust Cache omission policy based on local attention scores and robustness measures. Specifically, we compute the importance score of each in-cache token using averaged attention probability from future tokens, and formulate the eviction scope using tokens with the lowest variance of attention. RoCo exhibits significantly better consistency with full KV cache counterpart and robustness to eviction scope size.

To evaluate the effectiveness of RoCo in terms of preserving the LLM's downstream performance, we perform experiments across both prefilling and auto-regressive decoding stages at which cache eviction happens, spanning tasks of language modeling, text summarization, context reconstruction, and instruction following. Experimental results at different levels of KV cache budget demonstrates that RoCo results in significantly better generation quality compared to current methods judged by both automatic metrics and LLM-based evaluator.

Our contributions are summarized as follows:

- We systematically analyze current cache eviction policies from the dimensions of importance score calculation and eviction scope construction, shedding light on their limitations.
- Based on our analysis, we introduce a robust cache omission policy named RoCo and conduct a comprehensive evaluation to verify its effectiveness on downstream tasks.
- We open-source EasyKV, a versatile software package that supports key-value constrained generative LLM inference with flexible configuration on cache budget and eviction policy.

2 Background

In this section, we present necessary background about Transformer as well as existing literature on addressing the memory and computational bottleneck of Transformer-based LLMs.

2.1 Transformer-based LLMs

The input to a Transformer-based LLM is a sequence of tokens $\boldsymbol{x} = (x_1, ..., x_T)$, which is further processed by the embedding layer, followed by a series of Transformer blocks composed of an attention block and a feedforward block. The attention block is the only submodule where tokens at different positions exchange information, necessitating the need for a key-value cache during inference.

Attention Block At the l-th layer, the input hidden states $\boldsymbol{H}^{l-1} \in \mathbb{R}^{T\times d}$ is multiplied with three matrices $\boldsymbol{W}_q^l, \boldsymbol{W}_k^l$, and $\boldsymbol{W}_v^l$, producing $\boldsymbol{Q}^l = \boldsymbol{H}^{l-1}\boldsymbol{W}_q^l, \boldsymbol{K}^l = \boldsymbol{H}^{l-1}\boldsymbol{W}_k^l, \boldsymbol{V}^l = \boldsymbol{H}^{l-1}\boldsymbol{W}_v^l$. Then the scaled dot-product attention is performed as follows:

$$\text{Attn}_i = \text{Softmax}(\frac{\boldsymbol{Q}_i^l \cdot (\boldsymbol{K}_i^l)^\top}{\sqrt{d'}}) \cdot \boldsymbol{V}_i^l \quad (1)$$

$$\text{SDPA} = \text{Concat}(\text{Attn}_1, ..., \text{Attn}_H) \cdot \boldsymbol{W}_o^l \quad (2)$$

where H is the number of attention heads, $d' = \frac{d}{H}$ is the head dimension, and $\boldsymbol{W}_o^l$ is the output matrix.

Key-Value Cache LLM inference follows an autoregressive fashion. During training, it masks the upper triangular part of the attention matrix such that each token only sees itself and previous tokens. At inference time, the common practice is to cache the key-value vectors computed so far and append the newly computed ones into the cache. At time step T, the key-value cache can be written as a tensor of shape $(L, 2, B, H, T, d')$, where L is the number of model layers and B is the batch size. It is evident that the size of the KV cache grows linearly with respect to sequence length, potentially leading to excessive memory and latency issues when dealing with long input or output.

2.2 Efficient LLMs

Recent years have witnessed a surge of studies attempting to optimize the inference cost of LLMs from different (often orthogonal) perspectives.

One line of work follows the conventional model compression paradigm, aiming to identify and remove redundancy from billions of model parameters. These include tensor decomposition (Dao et al., 2022a), weight pruning (Frantar and Alistarh, 2023; Xia et al., 2023; Ashkboos et al., 2024), and quantization (Dettmers et al., 2022; Frantar et al., 2022; Xiao et al., 2023a). These methods reduce the KV cache footprint by reducing the model dimension, layers, and data precision.

Another line of work focuses on architectural design, aiming at reducing model complexity from the ground up. Representatives include sparse attention Transformers (Child et al., 2019; Zaheer et al., 2020), linear attention Transformers (Wang et al., 2020; Choromanski et al., 2020; Qin et al., 2022), and simplified attention variants (Shazeer, 2019; Ainslie et al., 2023). These methods either completely eschew the $O(T)$ space complexity of KV cache size or reduce the number of attention heads in exchange for a larger context length.

Some recent efforts (Liu et al., 2023b; Zhang et al., 2023; Oren et al., 2024) pay attention to methods that maintain the memory usage of the KV cache under a fixed budget without finetuning or architectural modifications to the model. The shared tenet of these approaches is the discernment and retention of key-value vectors that exert a significant influence on future generations. This work follows this line of research, dissects the efficacy of existing eviction policies, and introduces an improved policy with more consistent importance score and robust eviction scope construction.

3 Problem Formulation

Standard Inference Denote the input prompt to the LLM as $\boldsymbol{x} = (x_1, x_2, ..., x_T)$, the standard generative inference process consists of two consecutive stages: prefilling and auto-regressive decoding. The prefilling stage encodes the input prompt $\boldsymbol{x}$ and produces the corresponding attention key matrix $\boldsymbol{K}_T \in \mathbb{R}^{L\times H\times T\times d'}$ and value matrix $\boldsymbol{V}_T \in \mathbb{R}^{L\times H\times T\times d'}$, where L, H, d' represent the number of model layers, number of attention heads, and per-head dimension, respectively. Afterward, the LLM samples one token from its output distribution at each step conditioned on all key-value states computed so far. The key-value matrices are updated by appending the key-value vectors of this new token:

$$x_{T+1} \sim \text{LLM}(\cdot|x_{<=T}) \tag{3}$$

$$\boldsymbol{K}_{T+1} = \text{Concat}(\boldsymbol{K}_T, K_{T+1}) \tag{4}$$

$$\boldsymbol{V}_{T+1} = \text{Concat}(\boldsymbol{V}_T, V_{T+1}) \tag{5}$$

where $K_{T+1} \in \mathbb{R}^{L\times H\times 1\times d'}$, $V_{T+1} \in \mathbb{R}^{L\times H\times 1\times d'}$ are the key and value vectors of x_{T+1}. The above process is repeated until the end of sequence token is generated. Let $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{x}_o)$ denote the complete token sequence composed of input prompt $\boldsymbol{x}$ and output $\boldsymbol{x}_o$, where the output sequence contains N tokens. The peak cache size during standard inference is therefore determined by the key-value matrix $\{\boldsymbol{K}_{T+N}, \boldsymbol{V}_{T+N}\} \in \mathbb{R}^{L\times H\times (T+N)\times d'}$.

Key-Value Constrained Inference LLMs are typically deployed on hardware with constrained memory resources. However, during standard generative inference, the size of the key-value cache increases linearly with the total length of the sequence, potentially leading to out-of-memory issues and the associated latency incurred by reading and writing between High Bandwidth Memory (HBM) and Static Random Access Memory (SRAM) (Dao et al., 2022b).

To this end, recent studies have shifted toward key-value-constrained inference as a more controllable inference scheme. Denoting the fixed budget for each attention head as B tokens, key-value constrained inference is to maintain the key-value matrices $\boldsymbol{K}_t$ and $\boldsymbol{V}_t$ such that $\boldsymbol{K}_t, \boldsymbol{V}_t \in \mathbb{R}^{L\times H\times n\times d'}$ and $n \leq B$ for any $t \in \{1, ..., T+N\}$.

4 Eviction Policy for Key-Value Constrained Inference

In practice, $\boldsymbol{K}_t$ and $\boldsymbol{V}_t$ are stored in a fixed memory buffer with a maximum token budget B. When the buffer is full, an eviction policy is executed to remove stored but non-influential tokens from the cache. Although various eviction policies have been proposed, there still lacks a systematic comparison of their working mechanisms, design choices, and downstream performance.

To fill this gap, we embark on the efficacy of existing eviction policies from a unified framework. Concretely, we represent an eviction policy as the composition of two components: importance score calculation and eviction scope construction, which we elaborate on in the following sections.

4.1 Importance Score Calculation

Importance score calculation plays a vital role in eviction policy. It determines the relative order by which tokens are evicted. We summarize existing importance score calculation methods as follows:
Random Deletion As a naive baseline, one can randomly choose the key-value vectors to evict.

We incorporate this method into comparison and let it serve as the lower bound.

Recency This method deems the farthest token as least important and evicts it when the buffer is full. It is also referred to as window attention in prior studies (Ainslie et al., 2020; Beltagy et al., 2020; Xiao et al., 2023b).

Accumlative Attention Score (AAS) H_2O (Zhang et al., 2023) maintains a B-sized record array that stores the accumulative attention score each in-cache token received from subsequent tokens.

Accumlative Quantized Attention Score (AQAS) ScissorHands (Liu et al., 2023b) adopts an apporach similar to H_2O. The exception is that the attention score is quantized into a binary value, with 1 indicating above average and 0 indicating below average.

Last Token Attention Score (LTAS) TOVA (Oren et al., 2024) uses last token's attention score as importance indicator.

4.2 Eviction Scope Construction

Due to the auto-regressive nature of LLMs, recent tokens in the cache participate in less attention computation than earlier tokens. Therefore, their recorded importance scores for some attention-based methods can be underestimated and thus get wrongly evicted. To this end, an eviction scope should be constructed to carefully select tokens allowed to be evicted.

The dominant mean of constructing eviction scope is ***local window***, which assumes that tokens outside of a local window of size r have accumulated sufficient information on their importance.

4.3 Preliminary Experiments

In our controlled preliminary experiments, we are interested in how different importance score calculation methods behave in terms of consistency with respect to their full KV cache version. After that, we also explore another way of constructing eviction scope in addition to local window.

Setup We examine the Jaccard similarity between the top-B important tokens derived by various importance score calculation methods and those derived when a full KV cache is available. The higher the similarity, the more effectively the importance calculation method harnesses local information to approximate the global one. We evaluate all attention-based methods (i.e., AAS, AQAS, and LTAS) listed in Section 4.1 and set the local window size r to 0. More specifically, we use LLaMa2-7B-Chat[1] as the LLM and take the 805 instructions from AlpacaEval (Li et al., 2023) as prompt, generating a response for each instruction via greedy decoding. For KV cache budget from $\{0.3, 0.4, 0.5, 0.6\}$, we compute the Jaccard similarity at each token position, averaged over sequence length, attention heads, and layers.

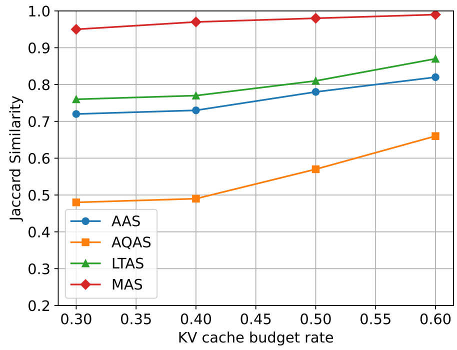

Figure 2: Consistency of different importance calculation methods w.r.t their full KV cache variant.

Results The results are shown in Figure 2. AAS shows a clear advantage over AQAS across all budget rates, indicating the importance of a full-precision attention score when the relative importance of tokens cannot be distinguished via binary value. LTAS has higher consistency than AAS, which we attribute to the fact that LTAS does not suffer from the recency bias that AAS and AQAS exhibit due to the accumulation operation. However, LTAS computes importance score based on a single token, which might bear high variability. Based on the above results, we advocate the use of *Mean Attention Score (MAS) to gauge the importance of each token*. MAS divides each token's accumulative attention score by how many times that token is attended by future tokens. As shown in Figure 2, MAS has remarkably higher consistency among all methods, achieving over 0.9 Jaccard similarity even at 0.3 cache budget rate. This verifies that MAS effectively alleviates recency bias and can better retain high-influential tokens.

Is Local Window the Optimal Way to Construct Eviction Scope? The local window approach has been widely used in conjunction with attention-based importance calculation methods to prevent recent tokens from being evicted. The underlying

[1] We also conduct experiments upon other LLMs like WizardLM-7B, and similar results are observed.

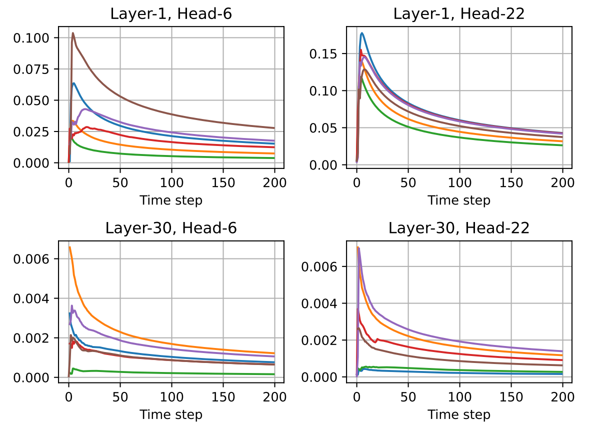

Figure 3: Illustration of persistence of attention robustness. We extract attention scores and compute the standard deviation from LLaMa2-7B-Chat.

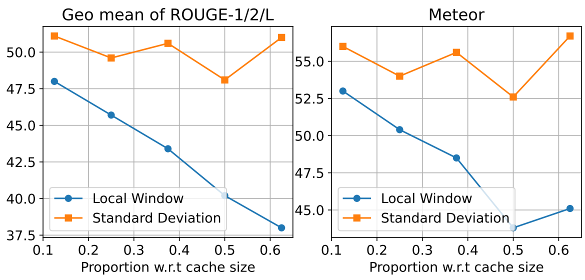

Figure 4: Results of MAS paired with local window and standard deviation on text summarization.

rationale is that the accumulated attention score is not sufficiently indicative until a specified threshold, i.e., the window size r is reached.

Based on the commendable consistency of MAS discussed in the previous paragraph, here we propose another way to construct the eviction scope which exploits a phenomenon termed *persistence of attention robustness* that we find ubiquitously exists in large language models. It states that the standard deviation of the attention probabilities a token receives from future tokens typically undergoes a brief ascending phase before settling into a stable decline, regardless of the model layer and attention head. Figure 3 clearly illustrates the phenomenon. We also observe that the ascending phase of a non-trivial portion of tokens only takes a relatively small number of steps, i.e., ≤ 50, suggesting it might be sub-optimal to only consider tokens at least r steps away for eviction scope construction.

To this end, we propose a new way to construct the eviction scope utilizing the *standard deviation* of attention scores. Concretely, we maintain another B-sized array for each attention head, keep track of the accumulative squared attention score, and compute the standard deviation of each in-cache token x using $\text{Std}(x) = \sqrt{\frac{\text{Acc}^{\text{square}}(x)}{\text{Count}(x)} - (\frac{\text{Acc}(x)}{\text{Count}(x)})^2}$. In practice, Acc, $\text{Acc}^{\text{sqaure}}$, and Count are all B-sized tensor and the standard deviation of all in-cache tokens are computed in parallel. Then, instead of the most recent r tokens, we exclude tokens having top-r standard deviation from eviction scope and remove the key-value vectors corresponding to the token with the lowest mean attention score.

We validate the effectiveness of both eviction scopes on a news summarization task with LLaMa2-7B-Chat and CNN/Daily Mail (See et al., 2017) dataset. Since summarization is a typical long-input-short-output task, we only perform KV eviction at the prefilling stage with a 0.5 compression rate and compare the output against the full KV cache version. Figure 4 shows the geometric mean of ROUGE-1/2/L (Lin, 2004) and METEOR (Banerjee and Lavie, 2005) for eviction scopes of different sizes. Standard deviation yields outputs with considerably higher quality while being less sensitive to the size of the eviction scope.

RoCo Combining mean attention score for importance score calculation and standard deviation for eviction scope construction, we introduce RoCo as a Robust Cache omission policy for key-value constrained generative LLM inference.

5 Comprehensive Evaluation

The goal of the eviction policy is to control the memory usage of key-value cache under a fixed budget while retaining the generation quality of LLMs as much as possible. In this section, we perform an empirical evaluation of the effectiveness of various eviction policies by taking the generated output with a full KV cache as the reference and comparing KV-restricted generations against it.

5.1 Experiment Setup

We describe the experimental setup used throughout our evaluation, including evaluation tasks, metrics, datasets, and compared eviction policies.

5.1.1 Tasks and Metrics

To broadly cover real-world use cases, we evaluate using four different types of tasks: language modeling, abstractive text summarization, original context reconstruction, and instruction following.

Language Modeling Language modeling task assesses the ability of LLMs to predict the next token given the preceding context. In the key-value-constrained scenario, a successful eviction policy should be able to detect and remove KV cache of unimportant tokens. Following prior works (Han et al., 2023; Xiao et al., 2023b; Oren et al., 2024), we adopt perplexity as the evaluation metric.

Abstractive Text Summarization Abstractive summarization requires extracting the most salient information provided in the input and generating a concise summary for it. Since the summary is usually much shorter compared to the input, we only perform cache eviction during the prefilling stage. We report BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and METEOR (Banerjee and Lavie, 2005) scores as the evaluation metrics.

Original Context Reconstruction Given the constrained incomplete key-value cache of an input document, the task of original context reconstruction measures how well the limited KV cache retains the essential information from the original context. BLEU and ROUGE scores are used as evaluation metrics.

Instruction Following Instruction following (Wei et al., 2021) requires an LLM to generate a proper response for a given user instruction. We apply KV cache eviction at the auto-regressive decoding stage since the model output tends to be more verbose. In addition to BLEU and ROUGE scores, we also opt for a pairwise comparison paradigm to evaluate the generated responses against those generated by text-davincci-003.

5.1.2 Datasets

We use the following datasets as the testbed for tasks described in Section 5.1.1.

OpenWebText OpenWebText is an open-source replication of the WebText dataset from OpenAI. We randomly sample 200 documents to form the test set for the language modeling task.

XSum Xsum (Narayan et al., 2018) comprises BBC articles from the years 2010 to 2017, encompassing a broad spectrum of topics.

CNN/Daily Mail CNN/Daily Mail (See et al., 2017) contains articles from the CNN and the Daily Mail newspapers, representing a different distribution from XSum. We use this dataset for both summarization and original context reconstruction.

AlpacaEval AlpacaEval (Li et al., 2023) is a model-based automatic evaluation benchmark for instruction-following LLMs. It comprises 805 instructions spanning a diverse range of scenarios.

5.1.3 Models

Following prior works (Zhang et al., 2023; Oren et al., 2024), we employ LLaMa2-7B-base for language modeling and LLaMa2-7B-Chat for the remaining tasks. We also include WizardLM-7B (Xu et al., 2023) as another strong instruction-tuned LLM for tasks except for language modeling.

5.1.4 Compared Eviction Policies

	Importance Score	Eviction Scope
Random	-	-
StreamLLM	-	local window
ScissorHands	AQAS	local window
H_2O	AAS	local window
TOVA	LTAS	-
RoCo	MAS	standard deviation

Table 1: Eviction policies considered in this paper. The definition of importance score and eviction scope are introduced in Section 4.1 and Section 4.2, respectively.

We consider the following baseline eviction policies, with their importance score calculation methods and eviction scope listed in Table 1:

- Random: evicting a randomly selected key-value pair from the cache.
- StreamLLM (Xiao et al., 2023b): evicting the key-value pair corresponding to the first token after 4 initial attention sink tokens.
- ScissorHands (Liu et al., 2023b): evicting the key-value pair corresponding to the token with the smallest accumulative quantized attention score outside the local window of size r.
- H_2O (Zhang et al., 2023): evicting the key-value pair corresponding to the token with the smallest accumulative attention score outside of the local window of size r.
- TOVA (Oren et al., 2024): evicting the key-value pair corresponding to the token with the smallest last token attention score.

5.1.5 Other Details

Our implementation is based on Pytorch (Paszke et al., 2019) and HuggingFace Transformers (Wolf et al., 2020). To improve the stability of outputs produced by LLMs, we employ greedy decoding for all generative tasks. For prefilling stage eviction,

Models	Methods	XSum					CNN/DM				
		BLEU	Meteor	R-1	R-2	R-L	BLEU	Meteor	R-1	R-2	R-L
LLaMa2-7B$_{\text{chat}}$	Random	16.7	35.1	41.8	22.8	33.3	15.4	30.8	39.0	19.9	26.2
	StreamLLM	11.9	35.5	42.7	17.6	31.3	19.2	40.0	49.5	24.3	30.7
	ScissorHands	29.3	50.4	56.5	35.6	46.5	27.8	46.4	57.7	33.4	40.7
	H_2O	37.3	55.8	62.2	44.2	54.2	31.1	47.6	58.9	36.8	43.7
	TOVA	20.5	42.4	48.9	26.9	39.9	21.5	41.7	53.1	27.4	34.8
	RoCo	**43.4**	**60.5**	**65.0**	**48.6**	**57.8**	**33.2**	**49.3**	**61.1**	**39.2**	**46.0**
WizardLM-7B	Random	12.6	30.0	36.3	19.1	28.0	12.4	27.8	39.6	17.0	23.9
	StreamLLM	7.1	27.3	36.2	13.4	25.6	8.8	25.9	39.0	14.4	23.8
	ScissorHands	29.8	48.6	57.2	38.5	47.7	24.6	41.2	54.3	29.1	35.7
	H_2O	30.5	50.2	57.6	40.8	49.7	27.7	43.3	56.4	32.5	39.2
	TOVA	15.0	36.2	46.1	23.9	34.8	14.5	32.7	45.9	19.4	27.9
	RoCo	**35.7**	**56.6**	**61.4**	**45.5**	**53.7**	**30.2**	**45.9**	**59.1**	**35.6**	**42.4**

Table 2: Performance of different eviction policies on abstractive text summarization tasks at 0.5 KV cache rate.

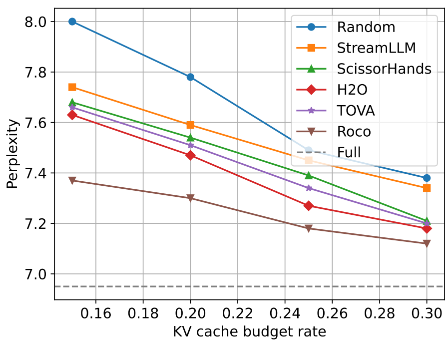

Figure 5: Performance of different eviction policies on language modeling task based on LLaMa2-7B.

Models	Methods	CNN/DM		
		BLEU	R-1	R-L
LLaMa2-7B$_{\text{chat}}$	Random	4.9	26.7	18.1
	StreamLLM	15.1	37.6	28.5
	ScissorHands	16.9	45.9	33.7
	H_2O	20.8	50.6	40.7
	TOVA	11.2	43.1	30.3
	RoCo	**28.1**	**57.7**	**49.2**
WizardLM-7B	Random	6.3	30.3	19.6
	StreamLLM	5.3	24.7	16.3
	ScissorHands	17.4	48.3	33.0
	H_2O	20.8	50.8	38.5
	TOVA	12.5	41.7	27.7
	RoCo	**29.0**	**58.9**	**47.5**

Table 3: Performance of different eviction policies on context reconstruction task at 0.5 KV cache rate.

we set budget size B by multiplying input token length with a compression rate (e.g., 0.5) and directly specify B as some integer for decoding stage eviction because the output length is unknown. The size of the local window is set to half of the KV cache budget following H_2O, i.e. $r = B/2$.

5.2 Main Results

Overview We report the results of RoCo alongside compared baseline eviction policies on language modeling, text summarization, original context reconstruction, and instruction following tasks in Figure 5, Table 2, Table 4, and Table 3. It can be seen that RoCo consistently outperforms previous methods by significant margins across all tasks and models. The advantage of Roco is particularly more evident at a low KV cache budget rate. As demonstrated in Figure 5, the gap between Roco and the second best method H_2O reaches 0.2 at 0.15 budget rate. We also notice that Roco yields larger improvements on generative inference tasks compared to language modeling. This is because, in generative tasks, future token predictions are dependent on previous model generations, where minor errors can accumulate and lead to large divergence. on the AlpacaEval benchmark, RoCo not only delivers the best overall performance in conventional metrics like BLEU and ROUGE, but also achieves comparable win rates as judged by GPT-4 (Table 5). In contrast, H_2O shows notable quality declines, and the gap is even larger for StreamLLM. Overall, the experiment results validate the effectiveness of Roco in terms of retaining model performance during key-value-constrained inference.

Attention Matters for KV Eviction As the only two policies that do not utilize attention-related information, Random and StreamLLM show significantly inferior performance compared to attention-based methods on all tasks. This observation aligns with the role of key-value vectors in attention computation, where the attention score serves natural indicator of token importance.

Models	Methods	AlpacaEval BLEU	ROUGE-1	ROUGE-2	ROUGE-L
LLaMa2-7B$_{\text{chat}}$	Random	45.6	66.6	49.7	56.1
	StreamLLM	47.3	66.6	51.1	57.1
	ScissorHands	62.1	76.5	63.9	68.8
	H_2O	63.0	77.5	65.7	69.9
	TOVA	60.1	75.3	63.8	68.3
	RoCo	**66.3**	**79.7**	**68.1**	**72.5**
WizardLM-7B	Random	40.6	62.9	45.6	52.8
	StreamLLM	43.0	63.9	47.8	54.5
	ScissorHands	57.2	73.3	60.9	65.3
	H_2O	58.6	74.1	62.1	66.8
	TOVA	59.6	75.0	63.1	68.0
	RoCo	**62.2**	**76.4**	**64.5**	**69.5**

Table 4: Performance of different eviction policies on AlpacaEval at 250-token KV cache budget.

Budget	StreamLLM	H_2O	RoCo
200	72.0(-3.6)	74.5(-1.1)	75.2(-0.4)
250	72.9(-2.7)	74.8(-0.8)	75.5(-0.1)

Table 5: AlpacaEval win rates against text-davincci-003 judged by GPT-4. Numbers in the parenthesis denote the performance drop compared to full KV cache (500 token output length on average with 75.6 win rate).

Block Size	BLEU	ROUGE-2	Speed up
1	41.2	46.2	1.0x
2	41.1	46.3	2.0x
4	41.1	46.2	4.0x
8	40.4	45.5	8.0x
16	40.2	45.4	16.0x

Table 6: Summarization results of block-wise eviction using RoCo.

5.3 Discussion

Extension to Grouped-Query Attention (GQA) GQA, along with its extreme case Multi-Query Attention (MQA) has gained increasing adoption in powerful LLMs like Mistral (Jiang et al., 2023). We extend the attention-based eviction policy to GQA and MQA by taking the group-wise averaged attention score and using it to update the importance score according to Section 4.1. The results of Zephyr-7B (Tunstall et al., 2023) in Appendix A validate the effectiveness of our extension.

Overhead An ideal eviction policy should avoid introducing much extra overhead since LLMs are already memory and computational-intensive. The memory overhead induced by Roco is $L \times H \times B \times 3$, which is negligible given the reduced KV cache footprint $L \times 2 \times H \times (S - B) \times d'$, where S is the non-evicted full sequence length. Moreover, different from the auto-regressive decoding stage which is I/O-bounded, the prefilling stage is computation-bounded. However, prior policies usually evict one token every time the cache is full and encode the next token, turning the prefilling stage into I/O-bounded. To accelerate key-value constrained prompt encoding, RoCo allows for performing evict-and-encode in a block-wise manner. The block size b controls the number of tokens being freed and encoded within one eviction step. We examine the effect of block-wise eviction using LLaMa2-7B-Chat on XSum summarization task. Table 6 shows that such block-wise eviction greatly speeds up prefilling while retaining similar output quality as token-wise eviction. More results are deferred to Appendix B due to space limit.

Integrated Package Finally, we open-source EasyKV, a software package dedicated to key-value constrained inference accompanying this research. It is designed to be fully compatible with existing LLMs with different attention variants and enables flexible configuration of diverse eviction policies, cache budgets, and application scenarios.

6 Conclusion

This paper studies key-value restricted language model inference. To shed light on the effectiveness of existing eviction policies, we conduct comprehensive comparative analysis by decomposing eviction policy into importance score calculation and eviction scope construction. We identify the inconsistency and instability of prior policie and introduce RoCo, a robust cache omission policy with improved downstream performance. We also release EasyKV, the accompanying library for versatile key-value constrained LLM inference.

Limitations

The first limitation of this work is its applicability to decoder-only Transformer models. For encoder-decoder style language models like T5 (Raffel et al., 2020), the KV cache in its encoder part involves bi-directional attention computation, which is not handled by existing cache eviction policies. We leave the adaptation to encoder-decoder models to future work as decoder-only LLMs are the mainstream and most capable models. Another limitation of this study lies in its practical implementation. The open-sourced implementation of this work is based on Pytorch and HuggingFace transformers library, which are not heavily optimized for GPU memory operation. Future iterations is dedicated to implementing cache operations with more efficient CUDA kernels.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. Etc: Encoding long and structured inputs in transformers. *arXiv preprint arXiv:2004.08483*.

Chenxin An, Shansan Gong, Ming Zhong, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

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

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. 2022a. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pages 4690–4721. PMLR.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022b. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Elias Frantar and Dan Alistarh. 2023. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*.

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. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023a. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023b. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *arXiv preprint arXiv:2305.17118*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

Matanel Oren, Michael Hassid, Yossi Adi, and Roy Schwartz. 2024. Transformers are multi-state rnns. *arXiv preprint arXiv:2401.06104*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Zhen Qin, Xiaodong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. 2022. The devil in linear transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7025–7041, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023a. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023b. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

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

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023. H _2 o: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv preprint arXiv:2306.14048*.

A More Experimental Results

Prompt Template For instruction-tuned/aligned LLMs used in the main experiments, we strictly follow their system prompt used during training. Specifically, we list the prompt template used for LLaMa2-7B/13B-Chat, WizardLM-7B, and Zephyr-7B as follows:

- LLaMa2-7B/13B-Chat: [INST] «SYS» You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information. «/SYS» {instruction}[/INST]

- WizardLM-7B: Below is an instruction that describes a task. Write a response that appropriately completes the request. Instruction: {instruction} Response:

- Zephyr-7B: <|system|> You are a friendly chatbot who always responds in a helpful and detailed manner to the user's questions.</s> <|user|> {instruction}</s> <|assistant|>

Extension to GQA/MQA We extend existing attention-based eviction policies into GQA and MQA by taking the group-wise averaged attention scores and using it to update the importance score. To verify our extension, we conduct experiments using Zephyr-7B (Tunstall et al., 2023), an instruction-tuned and aligned version of Mistral-7B (Jiang et al., 2023) on the text summarization task. Specifically, Zephyr-7B employs GQA and has 8 key-value heads and 32 query heads, rendering a 4x replication for each key-value vector pair.

The results are shown in Table 7. We can see that, attention-based eviction policies still exhibit better performance compared to Random and StreamLLM, showing that the group-wise average operation can effectively reflect the importance of each token across all query heads within its group.

Results on Larger LLMs In addition to 7B-scale LLMs, we also examine the effectiveness of RoCo alongside other eviction policies on 13B-scale LLMs. To this end, we conduct a text summarization experiment using LLaMa2-13B-Chat (Touvron et al., 2023b) and report the results in Table 8. We observe that, given the same KV cache budget, LLaMa2-13B-Chat achieves higher BLEU and ROUGE scores than LLaMa2-7B-Chat. It indicates that a larger model dimension may contain more redundancy in some less informative intermediate activations. This observation is inspiring because it implies that we can preserve more performance when using more powerful LLMs.

B More Results on Block-wise Eviction

To accelerate key-value constrained prompt encoding, we extend the per-token evict-and-encode scheme to a block-wise manner. We report results on text summarization with various block sizes using Zephyr-7B in Table 9. With a larger block size, more tokens are evicted with less reliable importance scores, thereby resulting in some influential tokens being wrongly evicted. Nevertheless, the performance drop is tolerable given the significant speedup of prompt encoding, especially when confronted with long-context tasks (Liu et al., 2023a; Bai et al., 2023; An et al., 2023).

C Case Study

To obtain a straightforward impression on the generation quality when RoCo is applied for key-value constrained inference, we present responses generated by LLaMa2-7B-Chat given the instruction "*What are the names of some famous actors that started their careers on Broadway?*". The responses at different KV cache budget are shown in Figure 6. At 0.3 KV cache budget rate, RoCo generates a response containing the same actor/actress names as the one conditioned on a full KV cache, demonstrating the commendable ability of RoCo to selectively retain useful key-value states and maintain coherent generation.

Models	Methods	XSum					CNN/DM				
		BLEU	Meteor	R-1	R-2	R-L	BLEU	Meteor	R-1	R-2	R-L
Zephyr-7B	Random	17.0	39.0	48.6	25.2	34.4	22.8	43.2	54.5	29.0	36.1
	StreamLLM	12.0	35.0	45.1	19.3	29.0	20.2	41.2	51.8	26.1	31.9
	ScissorHands	26.6	48.7	57.4	34.3	44.2	29.4	49.1	60.4	36.0	44.0
	H_2O	29.6	49.9	58.9	38.3	47.2	34.9	52.3	63.6	41.3	48.4
	TOVA	16.8	40.7	50.5	24.6	34.9	25.5	45.8	57.5	32.2	38.6
	RoCo	**33.6**	**54.9**	**62.6**	**42.4**	**50.4**	**36.6**	**53.8**	**64.6**	**42.6**	**50.0**

Table 7: Performance of Zephyr-7B using different eviction policies on abstractive text summarization tasks at 0.5 KV cache rate.

Models	Methods	XSum					CNN/DM				
		BLEU	Meteor	R-1	R-2	R-L	BLEU	Meteor	R-1	R-2	R-L
$\text{LLaMa2-13B}_{\text{Chat}}$	Random	14.9	27.7	30.0	17.9	24.5	9.0	18.8	22.3	11.7	15.1
	StreamLLM	14.7	35.7	41.7	19.8	31.2	26.0	43.1	53.2	30.4	36.6
	H_2O	38.3	55.9	61.0	44.5	53.4	38.4	52.5	62.3	42.6	48.6
	RoCo	**45.8**	**62.2**	**66.6**	**51.5**	**59.3**	**41.4**	**54.1**	**65.1**	**47.0**	**53.0**

Table 8: Performance of LLaMa2-13B-Chat using different eviction policies on abstractive text summarization tasks at 0.5 KV cache rate.

Instruction: What are the names of some famous actors that started their careers on Broadway?

Response of RoCo**,** 100% KV Cache:
Thank you for your question! There are many talented actors who have started their careers on Broadway and gone on to achieve great success in the entertainment industry. Here are some famous actors who got their start on Broadway:
1. Hugh Jackman: Known for his roles in "The Boy from Oz" and "The Greatest Showman," Jackman got his start on Broadway in the musical "Oklahoma!"
2. Audra McDonald: A six-time Tony Award winner, McDonald got her start on Broadway in the musical "Ragtime."
3. Idina Menzel: Menzel got her start on Broadway in the musical "Rent" and went on to star in "Wicked" and "Frozen."
4. Lin-Manuel Miranda: Miranda got his start on Broadway in the musical "In the Heights" and went on to create the hit musical "Hamilton."
5. Bernadette Peters: A legendary actress and singer, Peters got her start on Broadway in the musical "The Beautyful Nose" and has since starred in numerous productions, including "Gypsy" and "Sweeney Todd."
6. James Corden: Corden got his start on Broadway in the musical "Les Miserables" before becoming a late-night talk show host on "The Late Late Show with James Corden."
7. Christine Baranski: Baranski got her start on Broadway in the musical "The Producers" and has since appeared in numerous productions, including "The Good Wife" and "The Good Fight."
8. Nathan Lane: Lane got his start on Broadway in the musical "A Funny Thing Happened on the Way to the Forum" and has since starred in numerous productions, including "The Producers" and "It's Only a Play."
9. Bette Midler: Midler got her start on Broadway in the musical "Fiddler on the Roof" and went on to star in "Wicked" and "Hello, Dolly!"
10. John Leguizamo: Leguizamo got his start on Broadway in the play "A Day in the Death of Don Quixote" and has since appeared in numerous productions, including "The Lion King" and "Spike Lee's Summer of Sam."
I hope this helps! Let me know if you have any other questions.

Response of RoCo**,** 30% KV Cache:
Thank you for your question! There are many talented actors who have started their careers on Broadway and gone on to achieve great success in the entertainment industry. Here are some famous actors who got their start on Broadway:
1. Hugh Jackman: Known for his roles in "The Boy from Oz" and "The Greatest Showman," Jackman got his start on Broadway in the musical "Oklahoma!"
2. Audra McDonald: A six-time Tony Award winner, McDonald got her start on Broadway in the musical "Ragtime."
3. Idina Menzel: Menzel got her start on Broadway in the musical "Rent" and went on to star in "Wicked" and "Frozen."
4. Lin-Manuel Miranda: Miranda got his start on Broadway in the musical "In the Heights" and went on to create the hit musical "Hamilton."
5. Bernadette Peters: A legendary actress and singer, Peters got her start on Broadway in the musical "The Beautyful Nose" and has since starred in numerous Broadway productions.
6. James Corden: Corden got his start on Broadway in the musical "Les Miserables" before becoming a late-night talk show host on "The Late Late Show with James Corden."
7. Christine Baranski: Baranski got her start on Broadway in the musical "The Producers" before going on to star in the TV show "The Good Wife" and the movie "The Big Sick."
8. Nathan Lane: Lane got his start on Broadway in the musical "A Funny Thing Happened on the Way to the Forum" and has since starred in numerous Broadway productions, including "The Producers" and "The Birdcage."
9. Bette Midler: Midler got her start on Broadway in the musical "Fiddler on the Roof" before going on to star in the TV show "The Rose" and the movie "Hocus Pocus."
10. John Leguizamo: Leguizamo got his start on Broadway in the play "A Day in the Death of Jose Marti" before going on to star in numerous TV shows and movies, including "ER" and "Ice Age."
These are just a few examples of actors who got their start on Broadway. There are many other talented actors who have also gotten their start on the Great White Way.

Figure 6: Case study of LLaMa2-7B-Chat generated response given a specific instruction. The response generated with 30% KV cache using RoCo retains almost all content in the original response.

Block Size	BLEU	ROUGE-2	Speed up
1	26.5	35.4	1.0x
2	26.6	35.4	2.0x
4	26.5	35.4	4.0x
8	26.9	35.8	8.0x
16	26.4	35.4	16.0x

Table 9: Performance of Zephyr-7B on text summarization using RoCo. Larger block size only leads to a slight performance decline while significantly speed-up the prefilling stage.