CItruS 🍋: Chunked Instruction-aware State Eviction for Long Sequence Modeling

Anonymous ACL submission

Abstract

Long sequence modeling has gained broad interest as large language models (LLMs) continue to advance. Recent research has identified that a large portion of hidden states within the key-value caches of Transformer models can be discarded (also termed *evicted*) without affecting the perplexity performance in generating long sequences. However, we show that these methods, despite preserving perplexity performance, often drop information that is important for solving downstream tasks, a problem which we call *information neglect*. To address this issue, we introduce **C**hunked **I**ns**tru**ction-aware **S**tate Eviction (**CItruS**), a novel modeling technique that integrates the attention preferences useful for a downstream task into the eviction process of hidden states. In addition, we design a method for chunked sequence processing to further improve efficiency. Our training-free method exhibits superior performance on long sequence comprehension and retrieval tasks over several strong baselines under the same memory budget, while preserving language modeling perplexity.

1 Introduction

Recent advances in large language models (LLMs) have raised interest in long sequence modeling (Qin et al., 2023; Xiao et al., 2023). Several studies have found that information relevant to the next token prediction task often accumulates in the hidden representations of just a few tokens, and the attention distributions tend to focus sparsely on these tokens (Liu et al., 2024; Bai et al., 2024; Wang et al., 2023b). This observation has resulted in methods that model longer sequences by evicting unnecessary key-value caches during the language modeling process (Zhang et al., 2024b; Oren et al., 2024), mostly based on the attention weights each token receives from the following context.

However, these methods achieve limited performance on downstream tasks that require specific information from long documents (e.g., question answering), suggesting that they struggle to retain the detailed information necessary for such tasks. We refer to this condition as the *information neglect* problem. This issue arises because the cache acquired through state eviction is based only on the local document context. There is no explicit signal for the model to ensure that it is useful for solving downstream tasks. Consider Figure 1, which shows two attention distributions—one from a document context and one from an instruction prompt—when applying the Mistral 7B Instruct model to a sample from the Qasper dataset. Note that the two differ substantially in their weighting of positions, suggesting that the document context-derived attention weights may not capture well the task specified by the instructions.[1]

In this paper, we propose to address this information neglect issue by incorporating the instruction text into the state eviction process. Our method, **C**hunked **I**ns**tru**ction-aware **S**tate Evic-

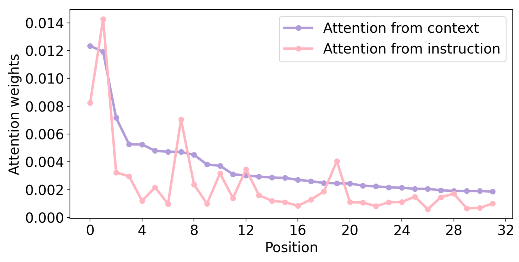

Figure 1: One sample from attention distributions in the 16th layer of the Mistral 7B Instruct model applied to the Qasper dataset. The attention distributions are calculated from a document context and an instruction text to the key-value cache. The x-axis represents different positions within the key-value cache, while the y-axis represents the attention weights. The positions are reordered by descending attention weights from the context, and positions with low attention weights are omitted for clarity.

[1]More details are provided in Section 3.

tion (**CItruS**), decomposes long sequence processing into two different subprocesses: language modeling and task solving. For the language modeling process, we propose chunked state eviction, splitting the long sequence input into large chunks while maintaining a cache that only stores the most important key-value states, which we show allows the model to efficiently and effectively encode long documents. As for the task-solving process, we propose an instruction-aware cache, either independent of or shared with the language modeling cache, which maintains the specific detailed information required to generate responses in downstream settings. The instruction-aware cache is then used to generate the final response for solving the task. Our approach can be seen as analogous to ideas from cognitive science that language and thought can be disentangled in human language processing (Fedorenko and Varley, 2016),

We evaluate CItruS on three tasks: long document reading comprehension, knowledge retrieval, and language modeling. Our approach improves downstream task performance over several strong baselines by large margins and enables the retrieval of desired information hidden within a long document of up to one million tokens. Furthermore, the model maintains high language modeling performance with a low perplexity. Notably, CItruS is applicable to all the transformer-based decoder-only model without any further training, improving the model's ability to conduct downstream tasks for input sequences with arbitrary lengths.

Overall, our contributions are summarized as follows: 1) We define and demonstrate the information neglect problem in state-eviction methods. 2) We propose CItruS, a state eviction method designed for long sequence downstream tasks, which incorporates an instruction-aware cache for task-solving and a chunked state eviction process for efficient language modeling. 3) Experiments on long document reading comprehension, knowledge retrieval, and language modeling show that CItruS improves performance on downstream tasks involving long sequence by large margins while maintaining low language modeling perplexity. Code and data will be released in the camera-ready version.

2 Related Work

2.1 Long Sequence Processing

Long sequence processing has long been a key research area in natural language processing (Tiezzi et al., 2024). Various approaches have been explored to address this challenge, including Longformer and State Space Models (Beltagy et al., 2020; Gu et al., 2022; Gu and Dao, 2023). Additionally, memory-augmented models use external memory to handle long sequences (Kuhn and De Mori, 1990; Wu et al., 2022; Bertsch et al., 2024; Lu et al., 2024), while recurrent-based transformers have been designed for long-sequence tasks (Dai et al., 2019; Li et al., 2023; Peng et al., 2023). More related work about long sequences is further provided in Appendix J.

Except for LONGHEADS, a memory-augmented method which requires storing all the past key-value states, all the above methods require further training of the model to handle long sequence processing. Our approach is an inference-time method and eliminates the need for further training, working directly with any open-source transformer-based language model and requiring significantly fewer resources than the methods mentioned.

Our work is also similar to retrieval-augmented generation (RAG) methods (Gao et al., 2023; Zhao et al., 2024), which incorporates knowledge from external databases to enhance the generation. However, RAG research mainly focuses on the retrieval process in order to better leverage the documents that could support the response generation, whereas CItruS is a method that more generally focuses on performing various long sequence tasks. It could be a good option to be applied to the RAG process. In fact, our testing includes long-document question answering and retrieval as primary tasks.

2.2 State Eviction for Large Language Models

Liu et al. (2024) explore the persistence of importance hypothesis for the key-value cache of large language models, which states that the position of the cache that are useful for language modeling tend to remain consistent over time. Based on this, various methods that evict the key-value cache during language modeling have been proposed for improving the efficiency of LLM inference. Zhang et al. (2024b) use accumulative attention scores to evict unnecessary key-value cache states. Oren et al. (2024) use the attention of the last token as a metric for evicting hidden states. Ge et al. (2023) profile all the attention heads and maintain different hidden states for different heads. Ren and Zhu (2024) propose determining the eviction scope by evaluating the standard variance of the attention weights received by individual tokens, and they test

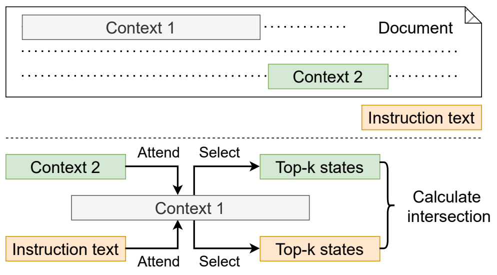

Figure 2: The illustration of our experiments that apply intersection calculation to explore the information neglect problem in state eviction models.

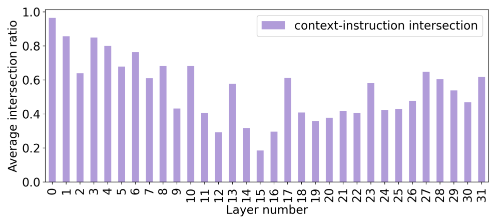

Figure 3: The difference between the top-k hidden states selected by the instruction text and the document context with the k set as 20, conducted with Mistral 7B Instruct. Context-instruction intersection represents the overlap between the top-k hidden states selected by the attention distribution from one piece of the context in the long document and the instruction text to a key-value cache.

the efficiency improvement of state eviction methods using small text chunks of size 16, which we scale up to 768 in our work. Yang and Hua (2024) bring the preference of future tokens into the state eviction process. Xiao et al. (2023) propose that "attention sinks" exist during LLM sequence processing. By keeping the key-value states of the initial tokens, and evicting the key-value states out of a sliding window maintained for recent tokens, their model could maintain the perplexity while processing 4 million tokens.

We propose that these previous methods suffer from the information neglect problem; that is, they fail to preserve specific information related to the instruction text, therefore might lower the performance on down-stream tasks.

3 The Information Neglect Problem

In this section, we demonstrate the information neglect problem of existing state eviction methods. State eviction methods have two basic elements: a key-value cache C that maintains the most important hidden states for language modeling and a strategy S to evict unnecessary states from the key-value cache, thereby making room for new states. By iteratively evicting the most unnecessary tokens from the cache, the model achieves the capability to model long sequences of arbitrary lengths. S is usually based on the attention weight a cache state receives from tokens later in the sequence.

The information neglect problem stems from the observation that the preserved states useful for language modeling are not necessarily the ones for a downstream task (e.g., answering a specific question). We demonstrate this by measuring the difference between the top-k states selected by a document context compared to those selected by a specific instruction text (Figure 2). Specifically, we select one context and encode it to acquire a cache that could be evicted (i.e, Context 1 in Figure 2). Then, we use another piece of context (i.e., Context 2 in Figure 2) and the instruction text, both with the same length, to evict the cache separately, retaining the top-k most important hidden states. By computing the overlap of the differently evicted caches, we draw conclusions about the information neglect scenarios during the eviction-based language modeling process. More experimental setup for these experiments is shown in Appendix A. We use the same setting to acquire the results in Figure 1.

We conduct this experiment on the full test set of the Qasper dataset (Dasigi et al., 2021). We use the averaged attention score of all the tokens from one piece of text to the cache to select the most important states, which is further described in Section 4.1. As shown in Figure 3, the hidden states focused on by the document context and the downstream instruction text are remarkably different, reflected by an intersection ratio lower than 0.2 in the middle layers.

Supported by the above experiments, we claim that if only the attention distribution of the context chunk is used to select the key-value states relevant to language modeling, some information specifically related to the final instruction text will be discarded during the encoding of the document, possibly decrease the task performance.

A similar line of work that models long sequence with sliding-window-based methods (Xiao et al., 2023; Han et al., 2023) also suffers from information neglect problems, where we provide detailed description in Appendix D.

4 Methods

To address the problem of information neglect, we propose to decompose the inference procedure of

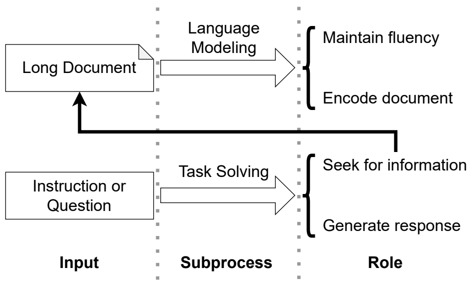

Figure 4: The illustration of our proposed different subprocesses for task-specific long sequence modeling. Each process serves as different roles.

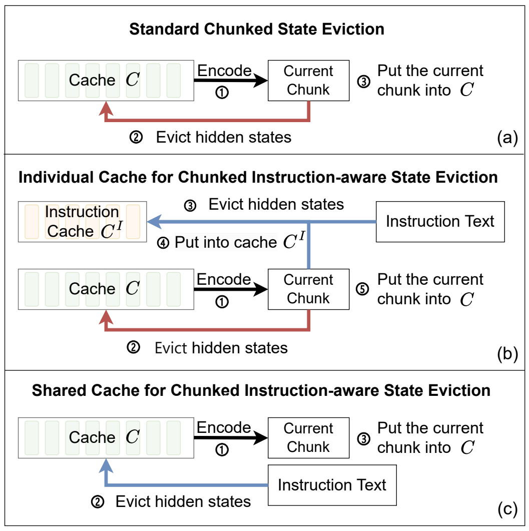

Figure 5: The illustration of different cache designs for our proposed Standard CSE and CItruS.

large language models into two different subprocesses: the language modeling process and the task solving process, shown in Figure 4. For the language modeling process, we propose to use **chunked state eviction** methods to make the modeling process more efficient. For the task solving process, we propose **instruction-aware state eviction**, using the hidden states of the final instruction prompt as an additional instruction-aware query to extract and preserve the task-related information in a key-value cache. Then, we utilize this key-value cache to generate a task-specific response.

For downstream tasks with a long document input D and a final instruction I (a piece of text that prompt the model to conduct the downstream tasks), our proposed method generates a corresponding response R according to I.

4.1 Chunked State Eviction (CSE)

In this section, we propose our standard state eviction method which chunks the input text during the language modeling process to enable the LLMs encoding the long document D more efficiently.

Overall process: Given a document D, we divide it into chunks $D = \{s_1, s_2, \ldots, s_n\}$, where n denotes the number of chunks. Each chunk s has a length of l_s except for the final chunk s_n. As illustrated in Figure 5(a), the Standard Chunked State Eviction (Standard CSE) process includes three steps: 1) given a cache C, we encode the current text chunk s with an LLM; 2) evict the unimportant hidden states in C according to the attention distribution from s to C; 3) put all the new hidden states of s into the cache. This iterative process starts with putting the first text chunk into the cache C and ends when the document has been fully processed. After the whole encoding process, the final chunk (maybe shorter than the length of l_s) is put into the cache, which leads to possible information bias towards this chunk. To alleviate this bias, we use the instruction text as a new text chunk to evict the cache C one more time. The resulting cache C is then used to encode the instruction text and generate the final response.

State eviction based on chunked averaged attention score: For state eviction, we use the attention score from all the tokens in the current text chunk s to a state c in the cache C as a metric of the state's importance:

$$\mathrm{Imp}(s, c) = \frac{1}{|s|} \sum_{t \in s} \frac{\exp\left(\frac{Q_t K_c^T}{\sqrt{d_k}}\right)}{\sum_{c' \in C} \exp\left(\frac{Q_t K_{c'}^T}{\sqrt{d_k}}\right)} \quad (1)$$

where $\mathrm{Imp}(s, c)$ represents the importance score of state c with chunk s, d_k is the dimensionality of the key vector, Q_t and K_c is the query vector of token t and the key vector of state c, respectively.

We preserve the states with the k highest importance scores while evicting the other states:

$$\mathbb{I}(x) = \begin{cases} 1, & \text{if } x \text{ in } \mathrm{Set}_{\text{selection}} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

$$\hat{C} = \{c \in C \mid \mathbb{I}(c) = 1\} \quad (3)$$

where $\mathrm{Set}_{\text{selection}}$ is the hidden states with the k highest importance scores $\mathrm{Imp}(s, c)$ from the current chunk s, and $\hat{C}$ represents the cache after the eviction. We execute the eviction in a layer-wise manner, which means that the hidden states retained in different layers could belong to different tokens. This design allows more flexibility since different layers could be responsible for different functions and semantics. We choose to not apply a finer-grained head-wise eviction to our model since it performed worse in our initial experiments.

4.2 Instruction-aware State Eviction

Next, we introduce chunked instruction-aware state eviction (CItruS) that aims to preserve information relevant to the task-solving process. We propose two kinds of cache design to achieve this goal. First, we propose to maintain a separate individual instruction cache C^I during the standard chunked state eviction process, which retains information related to the instruction text. Second, we propose a variant with a common shared cache for C^I and C to reduce the computational cost. Illustrations of the two proposed methods are shown in Figure 5.

Individual cache We use an individual instruction cache C^I to specifically store the hidden states related to the instruction text, in addition to C. Specifically, after the eviction on C, we conduct another eviction process on C^I with the final instruction text, and then put the key-value states of the current text chunk s into C^I. The eviction process is shown as follows:

$$\mathbb{I}^I(x) = \begin{cases} 1, & \text{if } x \text{ in Set}^I_{\text{selection}} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$\hat{C^I} = \{c^I \in C^I \mid \mathbb{I}^I(c^I) = 1\} \tag{5}$$

where $\text{Set}^I_{\text{selection}}$ is the key-value cache states with k highest importance scores of $\text{Imp}(I, c^I)$.

Shared cache Using individual caches will double the memory usage for a fixed cache size. Guided by the persistence of importance hypothesis (Liu et al., 2024), where the hidden states useful for maintaining the perplexity are attended by most of the following tokens, we hypothesize that the intersection between states selected by context and instruction texts, mentioned in Section 3, could be responsible for maintaining the perplexity. Hence, we suppose that we could further reduce the memory cost of C^I by sharing it with the language modeling process. Specifically, the top-k state $\text{Set}_{\text{selection}}$ of the shared cache is determined based on the attention-based importance score $\text{Imp}(I, c)$, which measures the attention from the final instruction I to a cache state c. Shown in Figure 5(c), we directly use this key-value cache evicted by $\text{Imp}(I, c)$ to encode the current chunk s. The rest of the eviction process follows the same procedure as described in Eq. (2) and (3).

4.3 Overall Process

In this section, we summarize the overall process for applying CItruS to downstream tasks. As described in Section 4.1, the model starts by iteratively encoding the chunked document D. Unlike the Standard CSE model, CItruS introduces the instruction text to evict either an individual or shared instruction-aware cache. As mentioned, we use the instruction text to evict these caches again after processing the entire document, selecting the k most important key-value states for each layer. We use these k states to encode the final instruction and generate the response, thereby setting the size of each cache for all models to k during this period[2].

5 Experimental Setup

5.1 Tasks

We compare the models using the following tasks. Detailed information about dataset statistics, prompts, and the divisions of document and instruction are provided in Appendices B and E.

Long document reading comprehension This task involves testing the ability of the models to answer a designated question based on a long document that exceeds the typical input length used during the pretraining of the large language models. In this task, we use the datasets of Qasper (Dasigi et al., 2021), MultifieldQA-en (Bai et al., 2023), HotpotQA (Yang et al., 2018), and TriviaQA (Joshi et al., 2017). We also include two other long few-shot tasks, Trec (Li and Roth, 2002) and SamSum (Gliwa et al., 2019), which focus on classification and dialogue summarization, respectively. We follow Bai et al. (2023) to adapt these datasets into long-document tasks. Instead of reporting the average scores in the main paper, we choose to report the average rank each model performs to avoid the variance differences among the datasets. Detailed results on each dataset is provided in Appendix C.

Long document knowledge retrieval We use two tasks to test if the model could preserve the important information during the whole language modeling process: passkey retrieval[3] (Mohtashami and Jaggi, 2023) and needle-in-a-haystack [4] tasks.

[2]The cache size of our standard CSE and shared cache CItruS during the encoding is $l_s + k$ while the individual cache CItruS requires a cache size of $2 \times (l_s + k)$.

[3]https://huggingface.co/datasets/lvwerra/needle-llama3-16x524k

[4]https://github.com/gkamradt/LLMTest_NeedleInAHaystack

The passkey retrieval task tests if the model can retrieve a single passkey (e.g., a five-digit number) inserted in a synthetic long document made up by repetitive simple sentences. We conduct this task on the documents with lengths up to 1 million tokens. The needle-in-a-haystack task replaces the passkey with a more general text fact and inserts them in real long documents. An example of the fact and the information of the documents can be found in Appendix F. The maximum length of documents for needle-in-a-haystack is set to 12,000. We use accuracy in the passkey retrieval task and the ROUGE metric (Lin, 2004) for the needle-in-a-haystack task to award partial correctness.

Long-range language modeling We report perplexity scores on long-range language modeling to estimate how well our models maintain fluency in generation (Xiao et al., 2023). We used PG19 (Rae et al., 2019) dataset.

5.2 Baselines

Streaming LLM always keeps the initial few tokens and uses a sliding window to model the long sequence (Xiao et al., 2023). This model is known for its ability of modeling long sequences with lengths up to 4 million tokens.

TOVA frames transformers as multi-state RNNs by using the attention distribution of the last token to identify which token should be evicted (Oren et al., 2024). This model could be seen as a special case of our standard CSE model with the l_s as 1.

H2O uses the accumulative attention score each token received to determine whether the token should be evicted (Zhang et al., 2024b).

RoCo uses averaged attention probability from future tokens and determines the eviction scope by evaluating the standard variance of the attention weights one token receives (Ren and Zhu, 2024).

LongHead (Lu et al., 2024) is another method that does not require further training. However, it requires large excessive memory cost (although could be offloaded to cpu memory, but that would cost more time) compared to our methods. Hence we choose to omit this model from our baselines to maintain a fair comparison.

Note that our proposed chunked instruction-aware state eviction is uncoupled with the eviction strategies used by the above models, hence it could be applied to all the above methods to achieve even better results. Due to the limitation of the computational cost, we only experiment the instruction-aware state eviction with our proposed chunked average attention score strategy and the accumulative attention score strategy used by H2O (denoted as H2O + Shared Cache) in our paper. All baselines are reimplemented with public repositories[5][6]. For all baseline models, we apply the same encoding and generation process described in Section 4.3 for fair comparison.

Settings	Mistral 7B Instruct			Llama 2 7B Chat			Llama 2 13B Chat		
	0-4k	4k-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+
Streaming LLM	2.83	3.17	3.50	2.50	3.00	4.17	1.67	3.17	3.83
TOVA	2.67	3.00	2.67	3.67	4.00	3.50	3.83	4.00	4.33
RoCo	3.67	2.67	2.83	3.00	3.17	2.00	4.00	1.33	2.33
H2O	4.00	3.50	4.17	4.17	2.50	2.67	3.33	3.50	4.83
Standard CSE	3.33	3.67	3.00	3.67	4.17	4.17	5.00	3.50	2.00
+ Individual Cache	**7.17**	**8.00**	**7.00**	6.50	7.00	6.67	6.50	7.33	6.33
+ Shared Cache	6.50	6.67	7.00	**6.83**	**7.33**	**7.33**	**6.50**	**7.33**	**6.33**
H2O + Shared Cache	5.17	5.17	5.50	5.33	4.83	5.50	4.67	5.17	5.83

Table 1: The averaged reversed rank results among all the 8 models on six different reading comprehension tasks, where 8 is the highest score and 1 is the lowest score. Results are presented by grouping text with lengths of 0-4k, 4k-8k, and 8k+. Best results are bolded.

5.3 Hyperparameters

We applied the position shift mechanism leveraged by Xiao et al. (2023), which always use the same positional embeddings for the caches containing different hidden states, to make the models process long documents better. We also apply this technique to all the baselines to enhance their ability of processing long sequences. We use the Llama 2 Chat model (Touvron et al., 2023) with 7 billion and 13 billion parameters and the 7 billion parameter Mistral Instruct model (Jiang et al., 2023) as the backbone models. k is set as 768 and l_s is set as 256, resulting a cache size of 1,024 during modeling the document. This setting is also applied to all the baseline models. We apply 8 bit quantization on the 13 billion parameter model. Results are inferred on one A100 80G GPU. All the hyperparameters are selected using the validation sets.

6 Results

6.1 Long document reading comprehension

The results of the long document reading comprehension tasks aggregated over six datasets are shown in Table 1, while the dataset-specific results are shown in Appendix C. First, our Standard CSE method achieves performance comparable to all the

[5]https://github.com/mit-han-lab/streaming-llm
[6]https://github.com/DRSY/EasyKV

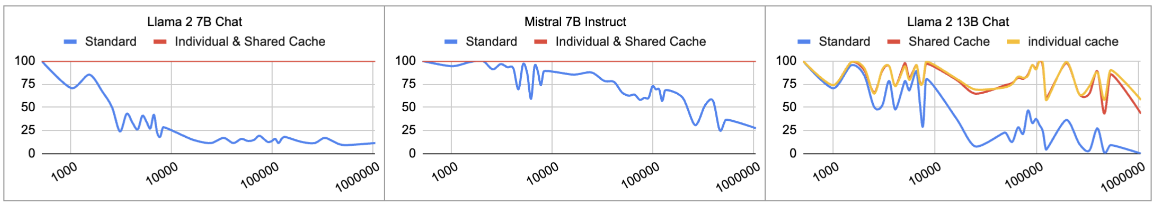

Figure 6: The results of the passkey retrieval task with Llama 2 7B Chat, Mistral 7B Instruct, and Llama 2 13B Chat.

Settings	Llama 2 7B Chat			Mistral 7B Instruct		
	R-1	R-2	R-L	R-1	R-2	R-L
$l_s = 256, k = 768$						
Standard CSE	19.87	5.74	17.52	15.17	6.34	13.94
+ Individual Cache	**24.72**	**7.87**	**24.53**	59.05	51.22	59.10
+ Shared Cache	23.73	7.46	23.44	**63.47**	**55.33**	**63.43**
$l_s = 1{,}024, k = 1{,}024$						
Standard CSE	18.86	7.52	18.04	30.21	14.18	29.23
+ Individual Cache	**33.28**	17.52	**32.76**	56.12	51.20	56.08
+ Shared Cache	31.95	**18.41**	31.47	**57.15**	**51.60**	**56.97**

Table 2: Results of the needle-in-a-haystack task. Best results are bolded. R-1, R-2, and R-L represent ROUGE-1, ROUGE-2, and ROUGE-L, respectively.

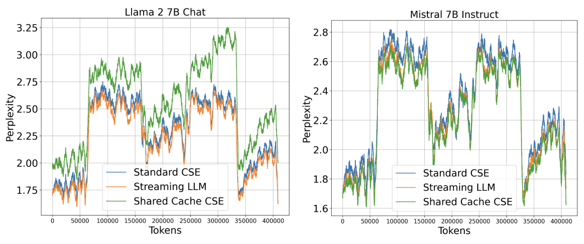

Figure 7: The language modeling results on the Llama 2 7B chat and Mistral 7B Instruct model. The line chart is smoothed with a window size of 4096 for clarity.

baselines, demonstrating the effectiveness of our basic framework. Both variants of CItruS consistently outperform all baselines and Standard CSE. As mentioned in Section 5.2, our method could also be applied on different eviction policies. Hence, we further included a variant of the H2O model (H2O + Shared Cache) and show that it achieves better performance over the H2O model in all cases.

We find models with a shared cache achieve the same level of performance as their corresponding model with separate caches. This suggests that the overlapping tokens between the context and the instruction text might be sufficient to support language modeling, while the shared cache also maintains the information useful for the downstream tasks. We will further discuss this in Section 6.3.

6.2 Long document knowledge retrieval

The main results of long document knowledge retrieval are shown in Figure 6 and Table 2. Our proposed CItruS retrieves all the passkeys using Llama 2 7B and Mistral 7B while still outperforming the Standard CSE for Llama 2 13B[7], which shows the superiority of CItruS for long document knowledge retrieval. For the needle-in-a-haystack task, our method outperforms the standard state eviction methods across different large language models and lengths.

6.3 Long-range language modeling

We compare our model with the long-range language modeling model, Streaming LLM. Specifically, we evaluate the standard CSE as well as the shared cache version of our proposed CItruS. For CItruS with a shared cache, we randomly sample 10 different instructions including different questions from Qasper and HotpotQA dataset. We show the results using one instruction here and append the rest of the results in the Appendix G. Results in Figure 7 show that our standard CSE could maintain the perplexity when processing long sequences as low as the Streaming LLM. Meanwhile, although showing a slight increase in perplexity with the Llama 2 7B Chat model, CSE with a shared cache achieves consistent perplexity results without exploding as described by Xiao et al. (2023). This shows that introducing the instruction text as the query to evict hidden states would not affect the text perplexity of the large language models. A more detailed discussion about the roles of the standard cache and the instruction-aware cache in our model is provided in Appendix H.

6.4 Analysis

In this section, we provide analyses on the hyperparameters of our model, the effect of chunk size, and the position bias in the knowledge retrieval tasks. We also provide an analysis on the effect of the initial tokens in Appendix I. We report the averaged results in this section since all the models

[7]We omit 5 outlier data points from all the 38 data points for Llama 2 13B Chat in the passkey retrieval tasks where all the models performs with an accuracy of 0%.

Param.	Settings	Llama 2 7B Chat			Mistral 7B Instruct		
		0-4k	4-8k	8k+	0-4k	4-8k	8k+
$l_s = 256$ $k = 768$	Standard CSE	36.72	37.07	38.36	34.52	30.57	20.92
	+ Individual Cache	**43.45**	43.26	45.93	**45.15**	**45.11**	**41.55**
	+ Shared Cache	43.22	**44.07**	**46.37**	43.96	41.56	36.61
$l_s = 512$ $k = 512$	Standard CSE	36.92	34.51	35.07	33.98	28.36	21.92
	+ Individual Cache	41.02	41.52	41.81	**45.13**	**43.38**	**41.66**
	+ Shared Cache	**41.17**	**41.79**	**43.57**	44.65	40.06	35.51
$l_s = 768$ $k = 256$	Standard CSE	32.59	31.04	29.57	30.60	23.19	21.44
	+ Individual Cache	34.73	33.79	33.88	**40.82**	**36.67**	**33.04**
	+ Shared Cache	**36.12**	**35.67**	**34.61**	38.89	32.50	28.77

Table 3: Results of the hyperparameters under the same memory budget. Best results are bolded. "Param." stands for hyperparameters.

Param.	Settings	Llama 2 7B Chat			Mistral 7B Instruct		
		0-4k	4-8k	8k+	0-4k	4-8k	8k+
$l_s = 64$ $k = 768$	Standard CSE	36.70	35.83	38.12	32.37	28.46	20.72
	+ Individual Cache	**43.78**	**43.86**	**47.48**	**45.88**	**44.01**	**40.20**
	+ Shared Cache	43.09	42.82	43.24	43.81	38.89	34.36
$l_s = 256$ $k = 768$	Standard CSE	36.72	37.07	38.36	34.52	30.57	20.92
	+ Individual Cache	**43.45**	43.26	45.93	**45.15**	**45.11**	**41.55**
	+ Shared Cache	43.22	**44.07**	**46.37**	43.96	41.56	36.61
$l_s = 512$ $k = 768$	Standard CSE	38.17	37.49	37.71	31.99	27.90	24.39
	+ Individual Cache	**43.04**	**43.18**	**46.71**	**42.79**	**42.09**	**40.58**
	+ Shared Cache	42.60	42.89	46.25	42.43	40.05	35.16
$l_s = 768$ $k = 768$	Standard CSE	39.41	38.25	38.39	33.35	25.74	20.51
	+ Individual Cache	**42.27**	**43.45**	42.26	42.31	**39.76**	**37.37**
	+ Shared Cache	42.09	42.61	**43.76**	**42.76**	38.72	33.56

Table 4: Results of the different chunk size l_s. Best results are bolded. "Param." stands for hyperparameters.

perform similarly in the analyses across all the datasets. The full results are shown in Appendix C.

6.4.1 Hyperparameter analysis

Given a fixed memory budget, there is a trade off between l_s and k. A larger k can preserve more information, potentially leading to a better performance, and l_s affects the encoding efficiency. In this section, we probe our model by adjusting different hyperparameters to demonstrate that our proposed CItruS is insensitive to them.

Table 3 shows that with a fixed budget, CItruS consistently outperforms the Standard CSE models, showing that our method is not sensitive to the choices of k and l_s, and the instruction-aware cache methods are the best when considering both the efficiency and the down-stream task performance.

6.4.2 Analysis of the chunk size

We provide a comparison of models using chunk sizes ranging from 64 to 768. The inference time of each model decreases linearly as l_s increases.

As shown in Table 4, the performance fluctuation when using different chunk sizes is very limited, while the efficiency is significantly improved. Our CItruS model extends the chunk size beyond that of previous methods and demonstrates a substantial improvement in efficiency for conducting long-sequence downstream tasks.

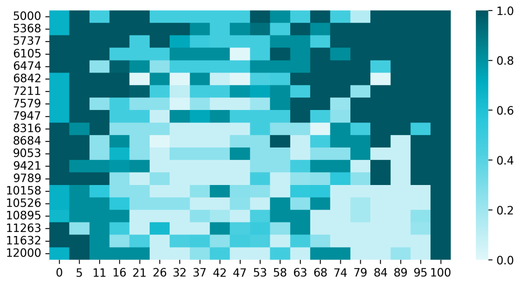

Figure 8: The position-wise results from CItruS with shared cache ($l_s = 1{,}024, k = 1{,}024$) on needle-in-the-haystack using Mistral 7B Instruct. The x-axis represents the position where the needle is inserted, while the y-axis represents the length of the documents. The color of the grid represents the ROUGE-1 score.

6.4.3 Position bias in knowledge retrieval

Liu et al. (2023) propose that large language models tend to pay less attention to the middle parts of long documents. In this section, we test our model to determine if this issue persists with our proposed instruction-aware cache method.

We use the needle-in-a-haystack task as the basic task and evaluate the ROUGE results when the fact is inserted at different positions in the document. As shown in Figure 8, we demonstrate that the CItruS model still prefers to attend to the information at the beginning and the end, leaving future work to address this lost-in-the-middle issue in eviction-based long-sequence methods.

7 Conclusion

We have proposed CItruS, an inference-time state eviction method for large language models (LLMs) that improves their performance on long sequence downstream tasks. It features a large chunked sequence processing procedure and an instruction-aware cache that helps with solving downstream tasks. Experiments on long document reading comprehension, knowledge retrieval, and language modeling show the utility of our method compared to strong baselines.

Our work demonstrates the possibility of generalizing standard LLMs trained on text constrained to certain lengths to processing longer sequences without any parameter adjustments. Our evaluation mainly focuses on retrieving task-related information from a long document. Future work may consider extending more high-level abilities (e.g., multi-hop and compositional reasoning) to the long sequence regime. Moreover, trainable components can be further introduced to facilitate this process.

Limitations

We only tested our methods with Llama 2 and Mistral models, leaving performance on other datasets to be evaluated. The instruction-aware cache is only applied to our Standard CSE and the H2O models, it could be further applied to models using other state eviction policies to possibly further enhance the performance. Our work only uses one instruction for each task to conduct all the experiments. It would be interesting to show whether better instruction texts exist that are specifically designed for conducting long sequence down-stream tasks. Future work might consider optimizing the query, or even use soft prompt optimization technique to select the hidden states.

Ethical Considerations

The associated risks with this work include using a model trained on vast amounts of text, which likely contains gender, racial, and cultural bias. Another concern is the potential misuse of the model for generating misleading or harmful content when applying our method to generate text. Meanwhile, cache-based methods could be more effective for malicious applications like jailbreaking or revealing private information, since it breaks the standard usage of the hidden states in large language models.

References

Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. 2024. Dynamic context pruning for efficient and interpretable autoregressive transformers. *Advances in Neural Information Processing Systems*, 36.

Yu Bai, Heyan Huang, Cesare Spinoso-Di Piano, Marc-Antoine Rondeau, Sanxing Chen, Yang Gao, and Jackie Chi Kit Cheung. 2024. Analyzing task-encoding tokens in large language models. *arXiv preprint arXiv: 2401.11323*.

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

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

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. 2024. Unlimiformer: Long-range transformers with unlimited length input. *Advances in Neural Information Processing Systems*, 36.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv: 2306.15595*.

Zihang Dai, Zhilin Yang, Yiming Yang, J. Carbonell, Quoc V. Le, and R. Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *Annual Meeting of the Association for Computational Linguistics*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. *North American Chapter of the Association for Computational Linguistics*.

Evelina Fedorenko and Rosemary Varley. 2016. Language and thought are not the same thing: evidence from neuroimaging and neurological patients. *Annals of the New York Academy of Sciences*, 1369(1):132–153.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv: 2312.10997*.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2023. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*.

Bogdan Gliwa, Iwona Mochol, M. Biesek, and A. Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *Conference on Empirical Methods in Natural Language Processing*.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv: 2312.00752*.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. *ICLR*.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Zero-shot extreme length generalization for large language models. *arXiv preprint arXiv: 2308.16137*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Dongseong Hwang, Weiran Wang, Zhuoyuan Huo, Khe Chai Sim, and Pedro Moreno Mengibar. 2024. Transformerfam: Feedback attention is working memory. *arXiv preprint arXiv:2404.09173*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv: 2310.06825*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Annual Meeting of the Association for Computational Linguistics*.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.

R. Kuhn and R. De Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. *arXiv preprint arXiv: 2402.09727*.

Xianming Li, Zongxi Li, Xiaotian Luo, Haoran Xie, Xing Lee, Yingbin Zhao, Fu Lee Wang, and Qing Li. 2023. Recurrent attention networks for long-text modeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3006–3019, Toronto, Canada. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

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, F. Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2024. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36.

Yi Lu, Xin Zhou, Wei He, Jun Zhao, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longheads: Multi-head attention is secretly a long context processor. *arXiv preprint arXiv:2402.10685*.

Hongyin Luo, Shang-Wen Li, Mingye Gao, Seunghak Yu, and James Glass. 2021. Cooperative self-training of machine reading comprehension. *arXiv preprint arXiv:2103.07449*.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *Neural Information Processing Systems*.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. 2024. Dynamic memory compression: Retrofitting llms for accelerated inference. *arXiv preprint arXiv: 2403.09636*.

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

Bo Peng, Eric Alcaide, Quentin G. Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, G. Kranthikiran, Xuming He, Haowen Hou, Przemyslaw Kazienko, Jan Kocoń, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, J. S. Wind, Stansilaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, P. Zhou, Jian Zhu, and Rui Zhu. 2023. Rwkv: Reinventing rnns for the transformer era. *Conference on Empirical Methods in Natural Language Processing*.

Guanghui Qin, Yukun Feng, and Benjamin Van Durme. 2023. The NLP task effectiveness of long-range transformers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3774–3790, Dubrovnik, Croatia. Association for Computational Linguistics.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Siyu Ren and Kenny Q Zhu. 2024. On the efficacy of eviction policy for key-value constrained generative language model inference. *arXiv preprint arXiv:2402.06262*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Matteo Tiezzi, Michele Casoni, Alessandro Betti, Tommaso Guidi, Marco Gori, and Stefano Melacci. 2024. On the resurgence of recurrent models for long sequences - survey and research opportunities in the transformer era. *arXiv preprint arXiv: 2402.08132*.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. 2023. Function vectors in large language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and

Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv: 2307.09288*.

Hongjie Wang, Bhishma Dedhia, and Niraj K. Jha. 2023a. Zero-tprune: Zero-shot token pruning through leveraging of the attention graph in pre-trained transformers. *arXiv preprint arXiv: 2305.17328*.

Junxiong Wang, J. Yan, Albert Gu, and Alexander M. Rush. 2022. Pretraining without attention. *Conference on Empirical Methods in Natural Language Processing*.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023b. Label words are anchors: An information flow perspective for understanding in-context learning. *Conference on Empirical Methods in Natural Language Processing*.

Jason Weston and Sainbayar Sukhbaatar. 2023. System 2 attention (is something you might need too). *arXiv preprint arXiv: 2311.11829*.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *arXiv preprint arXiv:2203.08913*.

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

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. 2023. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, R. Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *Conference on Empirical Methods in Natural Language Processing*.

Zi Yang and Nan Hua. 2024. Attendre: Wait to attend by retrieval with evicted queries in memory-based transformers for long context processing. *arXiv preprint arXiv: 2401.04881*.

Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. 2021. Tr-bert: Dynamic token reduction for accelerating bert inference. *arXiv preprint arXiv:2105.11618*.

Jungmin Yun, Mihyeon Kim, and Youngbin Kim. 2023. Focus on the core: Efficient attention via pruned token compression for document classification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13617–13628, Singapore. Association for Computational Linguistics.

Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu, and Zhangyang Wang. 2024a. Found in the middle: How language models use long contexts better via plug-and-play positional encoding. *arXiv preprint arXiv:2403.04797*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2024b. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36.

Jing Zhao, Junwei Bao, Yifan Wang, Yongwei Zhou, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2021. Ror: Read-over-read for long document machine reading comprehension. *arXiv preprint arXiv:2109.04780*.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv: 2402.19473*.

Yimeng Zhuang and Huadong Wang. 2019. Token-level dynamic self-attention network for multi-passage reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2252–2262, Florence, Italy. Association for Computational Linguistics.

A Details for the Intersection Probing Experiments

The goal of the intersection probing experiment is to determine whether the document context selects a different set of top-k states with the highest attention scores within the cache compared to the instruction text. This difference could lead to the document context overlooking crucial information required by the final instruction.

For this purpose, we use all the 416 documents in the test split of the Qasper dataset (Dasigi et al., 2021). For each document, we randomly select a chunk, referred to as Context 1, consisting of 200 tokens from the first $\frac{1}{4}$ document to simulate the cache C during the eviction process. If the first $\frac{1}{4}$ document contains fewer than 200 tokens, we use the entire first $\frac{1}{4}$ as Context 1. Then, we randomly select a second chunk, referred to as Context 2, from the final $\frac{1}{4}$ document to ensure sufficient distance between Context 1 and Context 2, avoiding recency bias and placing Context 2 close to the final instruction text. To ensure a fair comparison, we also make sure that the length of Context 2 is the same as that of the instruction text for each document.

We send the concatenation of Context 1 and Context 2 to the Mistral 7B Instruct model to obtain the simulated cache C, which consists of all the key-value states of Context 1. We could also acquire the attention distribution from Context 2 to Context 1 through this step. At each model layer l, we define the importance of the jth state in Context 1 as the average of a^l_{ij}, the attention score from each position i in Context 2 to the jth state in Context 1. We keep the top-k states in Context 1 with the highest average attention scores as $\text{Set}_{\text{selection}}$, and compute the final evicted cache $\hat{C}_{\text{context 2}}$ following Equ. (2) and (3). Similarly, we use the same model to encode the concatenation of Context 1 and the instruction text to get the attention distribution from the instruction text to Context 1, and follow the same steps as described above to obtain the final evicted cache $\hat{C}_{\text{instruction}}$ from the instruction text. In this experiment, we set k to 20, which is $\frac{1}{10}$ of length of the first context.

We compute the intersection ratio between the $\hat{C}_{\text{context 2}}$ and $\hat{C}_{\text{instruction}}$ as $\frac{|\hat{C}_{\text{context 2}} \cap \hat{C}_{\text{instruction}}|}{|\hat{C}_{\text{instruction}}|}$, and average the intersection ratio over all the 416 documents for each layer. As shown in Figure 3, the intersection ratio is particularly low in the middle layers of the model, supporting our hypothesis that the document context neglects a significant amount of information considered important by the final instruction. This discrepancy may be attributed to the remarkably different semantics of the instruction text and the document context, despite their close proximity.

B Statistics for Each Dataset

Qasper (Dasigi et al., 2021) consists of 5049 questions from 1585 NLP research papers. The questions are created by practitioners who read only the title and abstract, and answered by another group, who also provide supporting evidence. We use all available questions for each of the 224 documents selected by (Bai et al., 2023) from this dataset to evaluate model performance. When doing the intersection probing experiments, we use all 416 documents from the test split of Qasper. We randomly choose one question as the instruction text for each document.

MultifieldQA (Bai et al., 2023) consists of long articles from about 10 sources, including Latex papers, judicial documents, government work reports, and PDF documents indexed by Google. For each long article, several PhD and master students are invited to annotate. Each annotator is asked to propose questions with definitive answers as much as possible. We only use the English version of this dataset in our experiments. It contains 150 long documents.

HotpotQA (Yang et al., 2018) is a dataset with 113,000 question-answer pairs based on Wikipedia. This dataset requires multi-document reasoning to answer questions, and the questions are quite diverse and not tied to specific knowledge bases. HotpotQA has been adapted by (Bai et al., 2023) for long context evaluation, by concatenating the evidence text containing the answer along with several distracting articles. We use all 150 documents from the adapted HotpotQA for our experiments.

TriviaQA (Joshi et al., 2017) is a reading comprehension dataset containing over 650K question-answer-evidence triples. Averagely, six evidence documents are collected for each question. We use all 300 document-question pairs selected by (Bai et al., 2023), where each document consists of the concatenation of all available evidence documents for that question.

TREC (Li and Roth, 2002) is a question type

Settings	Llama 2 7B Chat			Llama 2 13B Chat			Mistral 7B Instruct		
	0-4k	4k-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+
Streaming LLM	33.78	34.92	37.11	37.39	37.95	36.54	34.26	31.00	27.21
TOVA	35.60	33.98	35.80	40.71	37.59	35.52	31.67	27.54	22.17
RoCo	32.46	25.70	20.64	40.30	31.02	25.89	32.67	25.28	19.83
H2O	34.47	29.54	27.26	38.68	35.24	35.96	34.60	25.37	23.08
Standard CSE	36.72	37.07	38.36	43.68	39.18	30.74	34.52	30.57	20.92
+ Individual Cache	**43.45**	43.26	45.93	46.43	46.61	40.80	**45.15**	**45.11**	**41.55**
+ Shared Cache	43.22	**44.07**	**46.37**	**46.66**	**46.91**	41.53	43.96	41.56	36.61
H2O + Shared Cache	38.26	39.19	40.27	42.00	41.54	**42.30**	40.28	36.23	32.45

Table 5: The averaged results on six different long sequence tasks. Results are separately presented by grouping text with different source lengths. Best results are bolded.

Models	Settings	Qasper			MultifieldQA			HotpotQA			Trec			TriviaQA			SamSum		
		0-4k	4k-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+
Llama 2 7B Chat	Streaming LLM	8.36	10.54	27.77	23.51	22.07	17.34	25.56	23.81	26.13	47.00	52.00	40.00	61.96	66.64	71.37	36.31	34.47	40.07
	TOVA	9.81	13.01	25.00	22.44	23.99	16.06	35.16	29.15	29.66	50.00	57.00	50.00	63.95	53.78	63.03	32.23	26.94	31.06
	RoCo	10.82	15.71	7.89	27.39	15.74	12.43	30.99	27.74	20.48	49.00	59.00	56.00	53.20	28.20	21.53	23.35	7.79	5.53
	H2O	9.31	12.61	14.54	31.08	21.66	23.52	46.92	35.22	32.74	50.00	54.00	43.00	62.45	48.06	45.76	7.06	5.67	4.00
	Standard CSE	8.43	14.84	27.08	23.19	22.67	15.06	33.30	24.68	35.27	49.00	55.00	52.00	70.10	75.05	71.99	36.27	30.17	28.77
	+ Individual Cache	20.71	16.80	28.77	36.65	28.69	24.67	43.04	38.95	41.75	55.00	62.00	65.00	67.44	80.30	82.10	37.87	32.84	33.26
	+ Shared Cache	20.78	17.48	29.02	38.43	30.81	24.90	43.21	40.59	42.88	56.00	63.00	63.00	64.14	80.05	83.50	36.74	32.48	34.93
	H2O + Shared Cache	16.91	14.87	33.52	40.59	31.80	25.78	44.00	38.32	36.07	45.00	53.00	49.00	75.27	73.26	66.71	7.76	23.88	30.55
Llama 2 13B Chat	Streaming LLM	12.48	12.86	19.81	24.36	24.61	14.22	28.66	30.12	31.88	52.00	58.00	44.00	76.09	77.91	75.22	30.76	24.19	34.12
	TOVA	17.18	13.76	23.20	26.75	24.34	14.23	40.79	29.56	36.59	56.00	59.00	49.00	80.05	84.22	76.43	23.51	14.66	13.69
	RoCo	16.09	12.86	24.41	26.85	17.48	10.08	39.09	26.26	15.84	57.00	58.00	45.00	82.41	69.71	59.27	20.34	1.79	0.72
	H2O	15.72	12.92	27.03	30.28	26.01	22.00	35.36	35.95	30.66	56.00	53.00	52.00	80.93	78.37	77.33	13.77	5.20	6.73
	Standard CSE	17.92	15.88	4.91	27.75	22.52	9.09	44.46	29.74	30.65	55.00	51.00	42.00	80.68	83.61	70.48	36.24	32.34	27.29
	+ Individual Cache	16.67	25.07	9.06	39.58	34.79	15.19	43.27	37.91	40.38	57.00	60.00	58.00	84.70	86.32	86.52	37.36	35.57	35.66
	+ Shared Cache	19.52	25.75	15.47	38.22	34.36	22.34	45.78	37.83	40.54	55.00	63.00	56.00	85.30	87.79	83.89	36.11	32.70	30.96
	H2O + Shared Cache	22.02	23.99	28.04	35.96	32.21	43.04	34.49	35.38	39.70	54.00	58.00	49.00	84.80	89.32	82.36	20.71	10.32	11.67
Mistral 7B Instruct	Streaming LLM	26.90	20.21	13.59	38.51	27.72	17.17	29.71	24.94	27.42	48.00	55.00	44.00	49.06	45.27	49.39	13.39	12.84	11.70
	TOVA	27.82	21.95	13.86	38.70	28.07	18.04	32.68	25.01	23.52	47.00	54.00	37.00	36.96	27.96	28.93	6.84	8.22	11.68
	RoCo	28.35	26.06	18.43	45.18	27.45	17.84	47.24	35.26	26.77	45.00	48.00	44.00	22.74	8.32	6.86	7.53	6.57	5.05
	H2O	27.02	22.67	14.60	48.68	32.12	31.21	49.16	35.36	31.83	48.00	49.00	47.00	21.97	3.00	1.00	12.76	10.06	12.85
	Standard CSE	27.73	21.08	7.61	42.01	28.17	19.78	37.74	34.01	27.14	46.00	55.00	33.00	31.16	21.36	16.02	22.50	23.79	21.97
	+ Individual Cache	29.93	27.66	14.93	55.10	40.75	45.48	45.88	46.10	32.07	50.00	64.00	57.00	61.99	61.69	65.64	27.99	30.44	34.19
	+ Shared Cache	30.93	27.14	19.18	54.34	40.53	45.96	45.71	45.37	35.53	50.00	59.00	52.00	56.19	48.16	35.69	26.61	29.15	31.30
	H2O + Shared Cache	29.41	23.47	18.85	53.28	38.04	42.23	45.99	45.70	34.21	48.00	62.00	52.00	57.83	43.32	42.67	7.17	4.87	4.72

Table 6: The detailed results on six different long sequence tasks, where $l_s = 256, k = 768$ for all methods. Results are separately presented by grouping text with different source lengths.

classification dataset collected from 4500 English questions published by USC (Hovy et al., 2001) together with 500 manually constructed questions for a few rare question types. This dataset has also been adapted for long context evaluation (Bai et al., 2023). This is achieved by sampling several cases from the training set to create few-shot examples as long context. We use all 300 examples from the adapted TREC.

SamSum (Gliwa et al., 2019) includes around 16K messenger-like conversations with summaries, created by English-fluent linguists. These conversations mirror the topics and styles of real-life messenger interactions, ranging from informal to formal, and may include slang, emoticons, and typos. Each conversation is annotated with a third-person summary, providing a concise overview of the discussion. This dataset has been adapted for long context evaluation as well in the same manner as the TREC dataset, and we use all 300 examples from this adaptation.

PG19 (Rae et al., 2019) includes a set of books extracted from the Project Gutenberg books library, that were published before 1919. We concatenate several selected books from this dataset to form a super long document and test the language modeling ability of our proposed methods on this document to up to 400K tokens in length.

C Detailed Results for Each Dataset

In this section, we provide the dataset results for all the experiments. In Table 5, we show the averaged results of all the baseline models and the CItruS model, while the detailed results containing different datasets are shown in Table 6. Dataset-wise experiment results using different hyperparameters are shown in Table 7, Table 8, Table 9. Table 10, Table 14, and Table 15.

Models	Settings	Qasper			MultifieldQA			HotpotQA			Trec			TriviaQA			SamSum		
		0-4k	4k-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+
Llama 2 7B Chat	Standard CSE	11.70	14.56	25.00	22.51	19.57	8.63	29.64	24.84	34.76	48.00	48.00	42.00	74.15	73.14	73.13	35.49	26.95	26.87
	+ Individual Cache	18.46	12.47	31.89	36.58	29.15	14.87	39.26	38.63	38.60	46.00	57.00	58.00	68.07	80.72	79.66	37.73	31.15	27.81
	+ Shared Cache	19.48	11.82	34.78	35.49	31.19	14.34	41.13	37.29	40.58	49.00	55.00	54.50	64.71	84.09	87.52	37.22	31.33	29.67
Llama 2 13B Chat	Standard CSE	17.57	12.26	3.13	29.76	15.33	6.94	37.94	23.75	13.13	52.00	47.00	41.00	84.62	69.46	41.09	34.39	20.98	13.63
	+ Individual Cache	19.29	23.11	1.69	38.49	20.53	10.49	38.26	26.90	22.27	53.00	59.00	55.00	85.48	79.79	81.49	34.71	28.20	20.33
	+ Shared Cache	20.36	21.80	5.90	38.89	21.18	12.75	40.20	27.66	22.26	54.00	61.00	49.00	85.39	78.56	69.72	33.15	24.04	14.97
Mistral 7B Instruct	Standard CSE	27.48	21.83	9.61	38.67	28.22	17.08	37.20	28.43	25.93	42.00	44.00	29.00	32.82	26.86	28.51	25.69	20.80	21.36
	+ Individual Cache	28.93	25.32	14.08	53.46	36.87	47.70	52.38	45.48	38.90	44.00	55.00	53.00	61.55	64.69	64.02	30.46	32.94	32.25
	+ Shared Cache	29.83	25.22	18.90	53.68	37.75	46.49	50.84	45.40	35.89	42.00	53.00	46.00	60.27	45.66	32.02	31.28	33.34	33.75

Table 7: The detailed results on six different long sequence tasks, where $l_s = 512, k = 512$ for all methods. Results are separately presented by grouping text with different source lengths.

Models	Settings	Qasper			MultifieldQA			HotpotQA			Trec			TriviaQA			SamSum		
		0-4k	4k-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+
Llama 2 7B Chat	Standard CSE	10.26	10.42	29.17	24.05	20.45	8.13	31.03	23.94	24.82	33.00	35.50	28.00	71.38	74.90	68.84	25.82	21.00	18.44
	+ Individual Cache	15.22	11.09	28.19	28.43	21.79	18.14	44.69	28.28	36.19	31.00	49.00	40.00	58.09	73.99	66.09	30.96	18.57	14.64
	+ Shared Cache	17.50	13.23	35.97	30.51	22.44	9.29	44.98	34.58	34.14	30.00	47.00	37.00	62.56	76.12	68.62	31.14	20.63	22.61
Llama 2 13B Chat	Standard CSE	11.76	4.74	5.72	21.47	8.90	5.64	19.25	4.87	11.98	41.00	41.00	28.00	69.39	35.10	35.48	24.52	3.98	2.67
	+ Individual Cache	18.02	10.93	12.26	32.66	20.36	15.97	28.30	23.30	26.32	44.00	48.00	41.00	84.61	78.51	81.02	26.34	7.97	7.29
	+ Shared Cache	17.27	10.20	10.85	37.06	22.92	17.46	26.75	26.69	26.45	42.00	49.00	37.00	84.08	78.79	76.02	26.76	8.72	6.33
Mistral 7B Instruct	Standard CSE	25.59	20.65	18.20	32.87	26.97	19.38	34.74	22.01	25.67	27.00	25.00	19.00	38.67	24.65	25.08	24.72	19.83	21.33
	+ Individual Cache	26.13	20.69	18.89	48.67	37.45	47.71	48.60	41.52	33.23	33.00	48.00	33.00	57.99	43.55	42.02	30.51	28.78	23.36
	+ Shared Cache	25.67	21.22	17.88	50.88	36.95	41.19	45.26	41.69	36.48	33.00	46.00	31.00	47.63	21.22	19.52	30.90	27.91	26.56

Table 8: The detailed results on six different long sequence tasks, where $l_s = 768, k = 256$ for all methods. Results are separately presented by grouping text with different source lengths.

D Information Neglect of the Sliding Window Methods

As pointed out by Jiang et al. (2023), the sliding window method with a window size of w would make the ith token representation h_i^l in a specific layer l access tokens from the input layer at a distance of up to $l \times w$. This is due to the inherent design of attention mechanism, where the representations of a former token in one layer could only be aggregated to the representations of a following token in the next layer. We describe this phenomenon more specifically by analyzing the equation of the sliding window attention mechanism for the token t_i with index i in a specific layer l,

$$a_{ij}^l = \frac{\exp\left(\frac{q_i^l \cdot k_j^l}{\sqrt{d_k}}\right)}{\sum_{j'=i-w}^{i} \exp\left(\frac{q_i^l \cdot k_{j'}^l}{\sqrt{d_k}}\right)} \tag{6}$$

$$\text{Attention}_i^l = \sum_{j=i-w}^{i} a_{ij}^l \cdot v_j^l \tag{7}$$

where d_k is the dimension of the hidden states, i and j are the the indexes of the query token and the tokens whose information are aggregated, respectively. As all the tokens are processed parallelly in one layer, the hidden states v_j and k_j could only contain their aggregated information from the previous layer, acquired by Attention_j^{l-1}. Considering q_i could only attend to $\{v_{i-w}, \ldots, v_i\}$ and $\{k_{i-w}, \ldots, k_i\}$ in one layer, the information aggregation range $r(i, j, l, l')$ for t_i from layer l' to l is,

$$r(i, j, l, l') = \bigcup_{l^*=l'}^{l} \left\{ \bigcup_{i^*=i-(l-l^*)\times w}^{i} \text{Attention}_{i^*}^{l^*} \right\} \tag{8}$$

Hence, the information of token t_i in the layer 0 (i.e., the embedding layer) would completely disappear in layer l after $l \times w$ time steps. Considering the effect that LLM would use specific layers to process the specifc information (e.g., syntax, task vector, etc) (Hendel et al., 2023; Todd et al., 2023), the specific information for one token might disappear merely after a few window lengths.

E Prompt for Each Task

We show all of the prompts we used for each task in Table 11.

F Setup for the Needle-in-a-Haystack Task

Due to the computational cost limitation, we used one fact to conduct this task. The fact is “The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day.” and the question input is “What is the best thing to do in San Francisco?”. The document is concatenated from documents from Paul Graham Essays. We cut the first 7 tokens where the model always generate

Models	Settings	Qasper			MultifieldQA			HotpotQA			Trec			TriviaQA			SamSum		
		0-4k	4k-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+
Llama 2 7B Chat	Standard CSE	10.15	13.93	25.00	25.91	21.59	16.65	32.68	31.54	33.44	55.00	56.00	51.00	71.75	73.64	73.72	33.52	28.26	26.42
	+ Individual Cache	19.97	14.91	29.33	32.87	28.94	24.18	43.58	39.46	42.37	53.00	61.00	64.00	68.72	81.19	85.60	37.44	31.83	32.00
	+ Shared Cache	19.63	15.07	30.28	33.19	30.11	27.17	41.13	38.12	43.18	57.00	60.00	63.00	69.41	82.80	86.93	37.90	33.00	29.70
Mistral 7B Instruct	Standard CSE	22.96	20.93	20.94	41.62	27.26	18.66	33.26	26.06	19.30	46.00	55.00	45.00	33.71	22.68	24.65	14.41	15.45	17.76
	+ Individual Cache	28.76	25.26	14.45	55.43	36.75	43.01	39.75	41.59	34.00	48.00	59.00	54.00	56.79	57.22	63.82	28.02	32.70	34.20
	+ Shared Cache	29.15	25.13	15.74	54.55	37.05	37.78	43.19	43.81	32.15	46.00	57.00	52.00	53.79	45.32	40.52	27.92	31.97	32.76

Table 9: The detailed results on six different long sequence tasks, where $l_s = 512, k = 768$ for all methods. Results are separately presented by grouping text with different source lengths..

Models	Settings	Qasper			MultifieldQA			HotpotQA			Trec			TriviaQA			SamSum		
		0-4k	4k-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+
Llama 2 7B Chat	Standard CSE	8.82	14.12	29.17	26.33	21.70	16.74	34.49	33.97	33.62	58.00	52.00	53.00	75.62	78.95	73.62	33.17	28.76	24.21
	+ Individual Cache	22.65	16.52	25.05	34.04	30.62	18.85	38.23	36.64	40.67	53.00	61.00	59.00	69.16	82.77	82.37	36.55	33.17	27.59
	+ Shared Cache	20.52	15.11	24.83	33.33	27.58	20.70	37.96	37.94	42.66	53.00	63.00	60.00	70.13	81.68	84.47	37.59	30.34	29.89
Mistral 7B Instruct	Standard CSE	26.24	22.10	17.73	42.75	26.37	16.64	35.25	26.72	19.11	48.00	51.00	38.00	31.18	15.87	16.21	16.66	12.35	15.37
	+ Individual Cache	28.28	23.91	16.08	54.69	36.15	35.06	41.04	38.40	29.53	45.00	59.00	54.00	56.79	50.32	56.02	28.03	30.79	33.54
	+ Shared Cache	28.99	23.95	15.14	55.93	37.74	35.83	42.36	39.44	29.61	47.00	56.00	52.00	53.79	44.82	39.69	28.46	30.34	29.11

Table 10: The detailed results on six different long sequence tasks, where $l_s = 768, k = 768$ for all methods. Results are separately presented by grouping text with different source lengths.

"The best thing to do in San Francisco is" to avoid the miscalculation of the information overlap. The template we used is shown in Table 11.

G Results of Perplexity with Other Instructions

We used 10 different instructions, shown in Table 12. We show the perplexity of models of CItruS with Shared cache when using these ten different instructions in Figure 9, Figure 10, Figure 11, and Figure 12. As these results demonstrate, the perplexity of our Shared Cache CSE remains consistent across a wide variety of instructions, similar to the standard CSE and streaming LLM methods.

H Discussion

In this paper, we argue that the cache used in standard chunked state eviction (CSE) is primarily responsible for maintaining the perplexity of language models, whereas an instruction-aware cache offers advantages for long-sequence downstream tasks. This claim is supported by the following observations from our experiments: (1) perplexity evaluations and previous work on state eviction methods (Zhang et al., 2024b; Oren et al., 2024) indicate that the basic cache effectively maintains language model perplexity; (2) performance improvements are observed when using an instruction-aware cache, which is only information that the model could access when generating the response during the task-solving thread. It is important to note that it is not solely the case that the standard cache only impacts perplexity while the instruction-aware cache solely affects task performance; there is potential overlapping, as demonstrated in our intersection calculation experiments discussed in Section 3. However, the primary focus of these two types of caches remains distinct.

I Analysis on initial tokens

Xiao et al. (2023) show that the initial tokens play a critical role in long-sequence language modeling by serving as "attention sinks". Although our proposed method does not specifically process the initial tokens, we assert that it can adaptively retain the hidden states of these tokens because they consistently receive a large proportion of attention weights. In this section, we conduct experiments that always preserve the first 4 initial tokens during the eviction process.

Shown in Table 13 and Table 15, we demonstrate that the difference between our methods with and without the initial tokens are limited, showing the capability of keeping the "attention sink" tokens using our method.

J More Related Work

J.1 Long Sequence Processing

Long sequence language modeling have attracted more and more research interests in recent years (Tiezzi et al., 2024). Various long document processing tasks are proposed to evaluate the long sequence modeling of language models (Zhao et al., 2021; Luo et al., 2021; Bai et al., 2023). Longformer, leveraging sparse self-attention pattern, save the memory cost to make the model process long document (Beltagy et al., 2020). Mem-

Datasets	Prompt
Qasper	You are given a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write “unanswerable”. If the question is a yes/no question, answer “yes”, “no”, or “unanswerable”. Do not provide any explanation.\n\nArticle: {context}\n\nAnswer the question based on the above article as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write “unanswerable”. If the question is a yes/no question, answer “yes”, “no”, or “unanswerable”. Do not provide any explanation.\n\n Question: {input}\n\n Answer:
MultifieldQA	Read the following text and answer briefly.\n\n{context}\n\nNow, answer the following question based on the above text, only give me the answer and do not output any other words.\n\nQuestion: {input}\n Answer:
HotpotQA	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nThe following are given passages.\n\n{context}\n\n Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: {input}\n Answer:
TriviaQA	Answer the question based on the given passage. Only give me the answer and do not output any other words. The following are some examples.\n\n\n\n{context}\n\n\n\n Question: {input}\n\n\n\nAnswer:
TREC	Please determine the type of the question below. Here are some examples of questions.\n\n\n\n{context}\n\n{input}
SamSum	Summarize the dialogue into a few short sentences. The following are some examples.\n\n\n\n{context}\n\n\n\n{input}
Passkey Retrieval	There is an important info hidden inside a lot of irrelevant text. Find it and memorize them. I will quiz you about the important information there.{context}\n\n\n\nWhat is the pass key? The pass key is
needle-in-a-haystack	system: You are a helpful AI bot that answers questions for a user. Keep your response short and direct \n\n user: {context}\n\nuser: {Question} Don’t give information outside the document or repeat your findings\n\n system:

Table 11: The prompt used in our experiments. Text in blue represents the context while text in red represents the instruction we used.

Datasets	Prompt
Instruction 1	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: How is the ground truth for fake news established?\nAnswer:
Instruction 2	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: What architecture does the encoder have?\nAnswer:
Instruction 3	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: Which case was brought to court first Miller v. California or Gates v. Collier ?\nAnswer:
Instruction 4	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: What occupation is shared by both Marge Piercy and Richard Aldington?\nAnswer:
Instruction 5	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: What is their definition of tweets going viral?\nAnswer:
Instruction 6	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: Were any of these tasks evaluated in any previous work?\nAnswer:
Instruction 7	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: What sentiment classification dataset is used?\nAnswer:
Instruction 8	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: The historical Nimavar school in the Nimavar Bazaar, or bazar, is located in which country?\nAnswer:
Instruction 9	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: For what type of work is the production company for The Year Without a Santa Claus best known?\nAnswer:
Instruction 10	Answer the question based on the given passages. Only give me the answer and do not output any other words.\n\nQuestion: The physicist who is responsible for identifying the Rabi cycle won what award?\nAnswer:

Table 12: The instruction used in the perplexity experiments.

Param.	Settings	Llama 2 7B			Mistral 7B		
		0-4k	4-8k	8k+	0-4k	4-8k	8k+
Start size = 0	Standard CSE	36.72	37.07	38.36	34.52	30.57	20.92
	+ Individual Cache	**43.45**	43.26	45.93	**45.15**	**45.11**	**41.55**
	+ Shared Cache	43.22	**44.07**	**46.37**	43.96	41.56	36.61
Start size = 4	Standard CSE	36.30	34.80	37.42	31.44	28.51	21.10
	+ Individual Cache	**43.48**	**43.89**	46.36	**45.69**	**44.55**	**42.22**
	+ Shared Cache	43.44	43.65	**46.97**	44.22	41.74	36.05

Table 13: Results of the different start sizes averaged on six different long sequence tasks. Best results are bolded. “Param.” stands for hyperparameters.

orizing transformer uses a external memory to save the information during the long sequence modeling process (Wu et al., 2022). Mistral applied Pre-fill and chunking sliding window methods to model longer sequences (Jiang et al., 2023). State space models and their variations are also popular recently (Gu et al., 2022; Gu and Dao, 2023; Wang et al., 2022). Unlimited-former wraps pretrained encoder-decoder transformer, and offloads the cross-attention computation to a single k-nearest-neighbor index, while the returned kNN distances are the attention dot-product scores (Bertsch et al., 2024). Nawrot et al. (2024) propose to compress the key-value cache to make the model process longer sequences. Xiong et al. (2023) conduct continual pretraining from Llama 2 (Touvron et al., 2023) with longer training sequences and on a dataset where long texts are upsampled. Rotary Position Embedding and the positional interpolation based on it are also used enable the model process longer sequences (Su et al., 2024; Chen et al., 2023). ReadAgent are proposed by using a large language model agent to process long sequences (Lee et al., 2024). LONGHEADS enhances the long-context processing of large language models by allowing multi-head attention to attend to selected important context chunks within the trained length (Lu et al., 2024). Infini-Transformer leverage a compressive memory between different context segment to achieve modeling long range text (Munkhdalai et al., 2024). Hwang et al. (2024) propose TransformerFAM, a novel architecture with a feedback loop for attending to latent representations, enables Transformers to process indefinitely long sequences without additional weights. Zhang et al. (2024a) leverage plug-and-play positional encoding to make the model better collect the information in the middle of the document.

Except LONGHEADS which requires storing all

Models	Settings	Qasper			MultifieldQA			HotpotQA			Trec			TriviaQA			SamSum		
		0-4k	4k-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+
Llama 2 7B Chat	Standard CSE	10.30	12.43	25.00	23.86	21.75	14.50	34.14	28.58	34.09	47.00	50.00	42.00	68.88	71.73	79.50	36.00	30.49	33.61
	+ Individual Cache	20.43	18.82	29.64	39.99	32.17	37.79	43.87	36.28	43.36	57.00	64.00	63.00	63.32	77.14	77.19	38.06	34.72	33.89
	+ Shared Cache	20.14	18.07	27.49	40.00	33.69	37.06	43.47	41.77	44.26	56.00	63.00	55.00	61.42	66.59	58.91	37.50	33.81	36.72
Mistral 7B Instruct	Standard CSE	24.95	19.05	8.49	36.36	27.86	17.88	34.95	27.88	23.47	47.00	46.00	28.50	33.18	34.75	34.96	17.75	15.24	11.02
	+ Individual Cache	32.00	28.45	15.80	57.42	39.76	47.02	47.02	41.63	30.00	51.00	64.00	56.00	59.43	61.32	67.01	28.40	28.90	25.38
	+ Shared Cache	32.08	24.10	19.48	56.78	39.69	47.58	45.67	50.20	38.66	51.00	61.00	58.00	54.19	31.49	24.52	23.12	26.83	17.90

Table 14: The detailed results on six different long sequence tasks, where $l_s = 64, k = 768$ for all methods. Results are separately presented by grouping text with different source lengths.

Models	Settings	Qasper			MultifieldQA			HotpotQA			Trec			TriviaQA			SamSum		
		0-4k	4k-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+	0-4k	4-8k	8k+
Llama 2 7B Chat	Standard CSE	9.27	11.22	25.00	26.15	23.38	15.26	31.11	31.17	33.44	49.00	53.00	54.00	67.46	61.89	65.32	34.83	28.11	31.52
	+ Individual Cache	21.32	17.49	28.47	37.21	29.16	28.08	42.19	39.02	40.40	54.00	65.00	65.00	69.07	80.10	81.60	37.06	32.56	34.61
	+ Shared Cache	21.33	15.70	33.19	38.19	30.76	25.46	44.36	37.93	42.21	56.00	65.00	64.00	64.14	80.05	82.28	36.60	32.43	34.66
Mistral 7B Instruct	Standard CSE	25.91	21.99	9.25	41.60	28.47	19.17	32.04	28.17	21.81	44.00	47.00	34.00	29.55	30.22	27.04	15.52	15.22	15.35
	+ Individual Cache	30.37	27.09	15.30	56.32	40.28	47.21	47.74	45.98	33.80	49.00	64.00	56.00	61.99	59.69	66.14	28.73	30.26	34.87
	+ Shared Cache	30.39	25.59	19.49	54.85	40.90	44.92	44.88	44.63	36.72	51.00	62.00	51.00	57.43	47.16	35.19	26.78	30.17	28.97

Table 15: The detailed results on six different long sequence tasks, where the start size is set to 4 and $l_s = 256, k = 768$ for all methods. Results are separately presented by grouping text with different source lengths.

the past key-value states, all the above needs further training to make the model able to handle the long sequence processing task. Our work do not need any training and can be applied directly to any open-source transformer-based large language models.

J.2 State Eviction for Large Language Models

Liu et al. (2024) explore the persistence of importance hypothesis for the key-value cache of large language models. They establish that the key-value cache that useful for large language modeling are consistent for all the following text. Based on this, various methods that evicts the key-value cache during the language modeling has been proposed for improving the efficiency of the LLM inference. Xiao et al. (2023) propose that "attention sink" exists during the sequence processing of large language models. By keeping the key-value states of the initial tokens, and evict the key-value states out of a sliding window maintained for recent tokens, the model could maintain the perplexity while processing 1 million tokens. Zhang et al. (2024b) use accumulative attention scores to evict the unnecessary key-value cache states. Oren et al. (2024) uses the attention of the last token as a metric to evict the hidden states. Ge et al. (2023) profile all the attention heads and maintain different hidden states for different heads. Attendre (Yang and Hua, 2024) brings the preference of future tokens into the state eviction process.

Besides inference-only state-eviction, a lot of methods also explore to learn to prune tokens during the training process in computer vision (Wang et al., 2023a; Kim et al., 2022; Ye et al., 2021) or natural language processing (Zhuang and Wang, 2019; Frantar and Alistarh, 2023; Yun et al., 2023; Anagnostidis et al., 2024). There is also work that delete tokens from the discrete prompt (Weston and Sukhbaatar, 2023).

Compared to this paper, the previous work rarely focuses the state eviction technique on the long sequence modeling scenario and does not related to the specific optimization for the down-stream tasks.

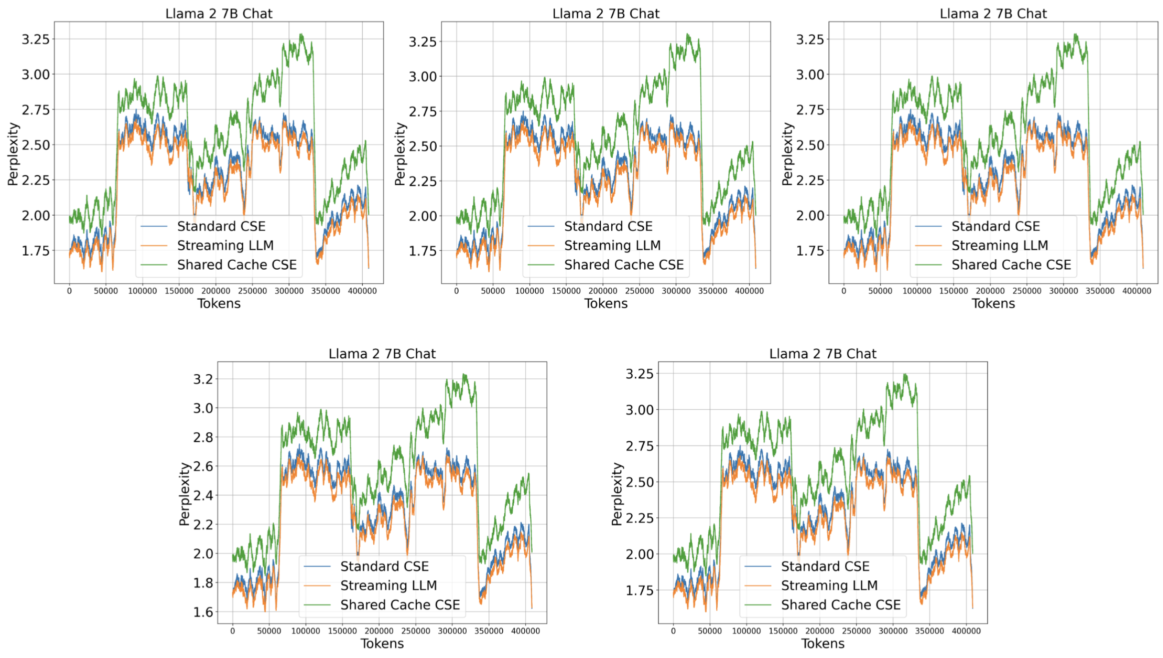

Figure 9: The language modeling results on the Llama 2 7B chat model. The instructions 1 to 5 listed in table 12 are used for the Shared Cache CSE method, respectively. The line chart is smoothed with a window size of 4096 for better visibility.

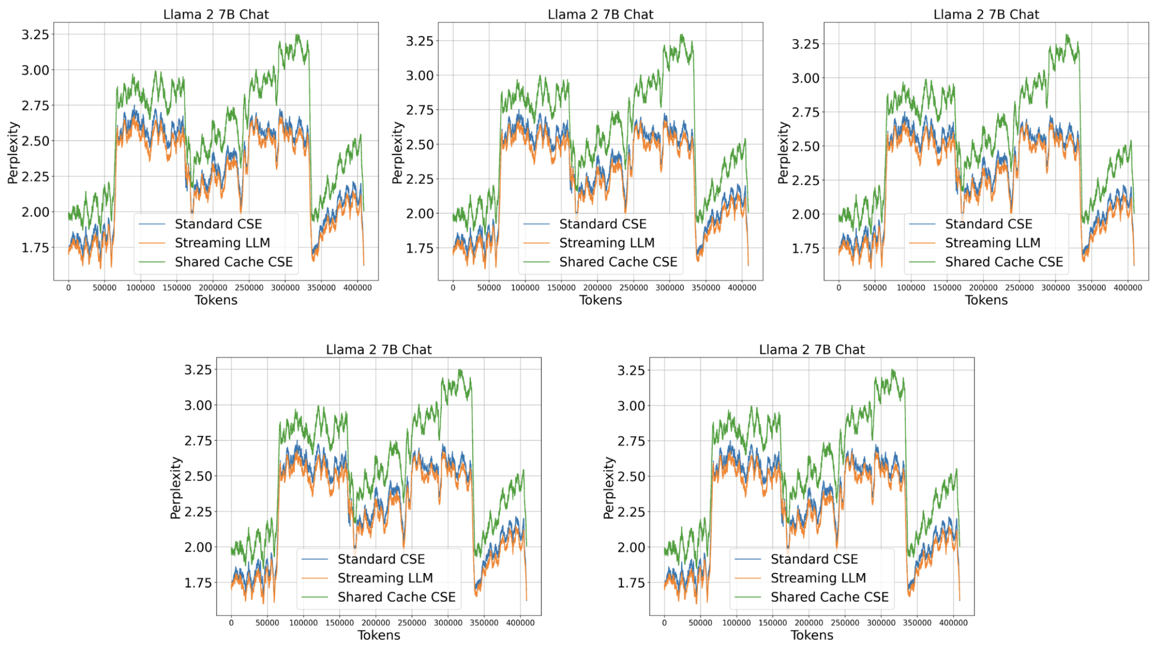

Figure 10: The language modeling results on the Llama 2 7B chat model. The instructions 6 to 10 listed in table 12 are used for the Shared Cache CSE method, respectively. The line chart is smoothed with a window size of 4096 for better visibility.

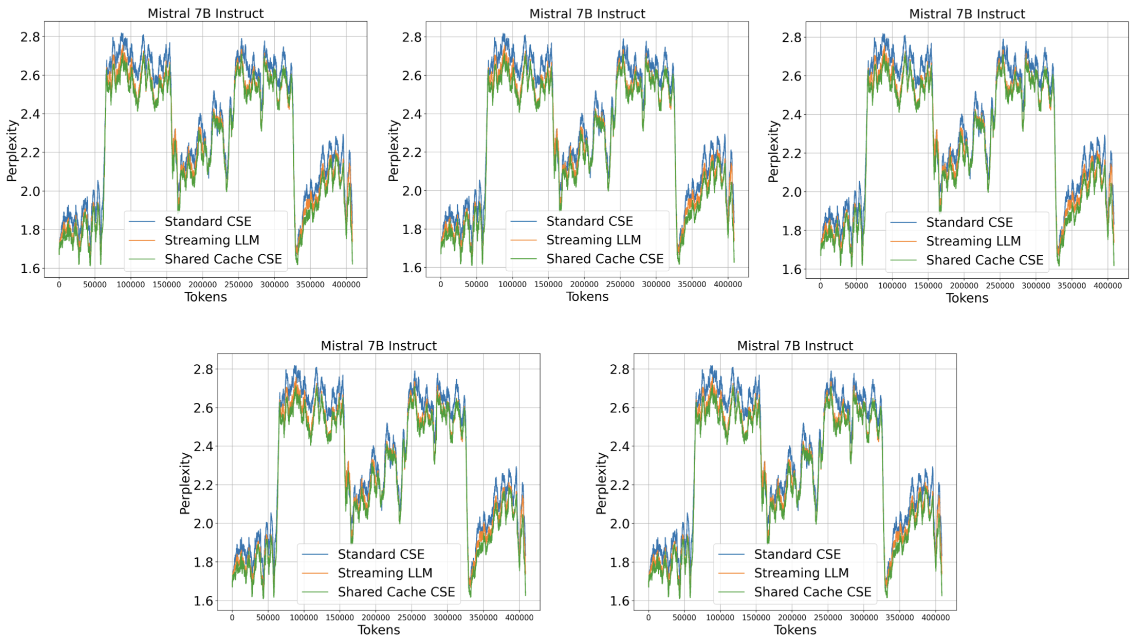

Figure 11: The language modeling results on the Mistral 7B Instruct model. The instructions 1 to 5 listed in table 12 are used for the Shared Cache CSE method, respectively. The line chart is smoothed with a window size of 4096 for better visibility.

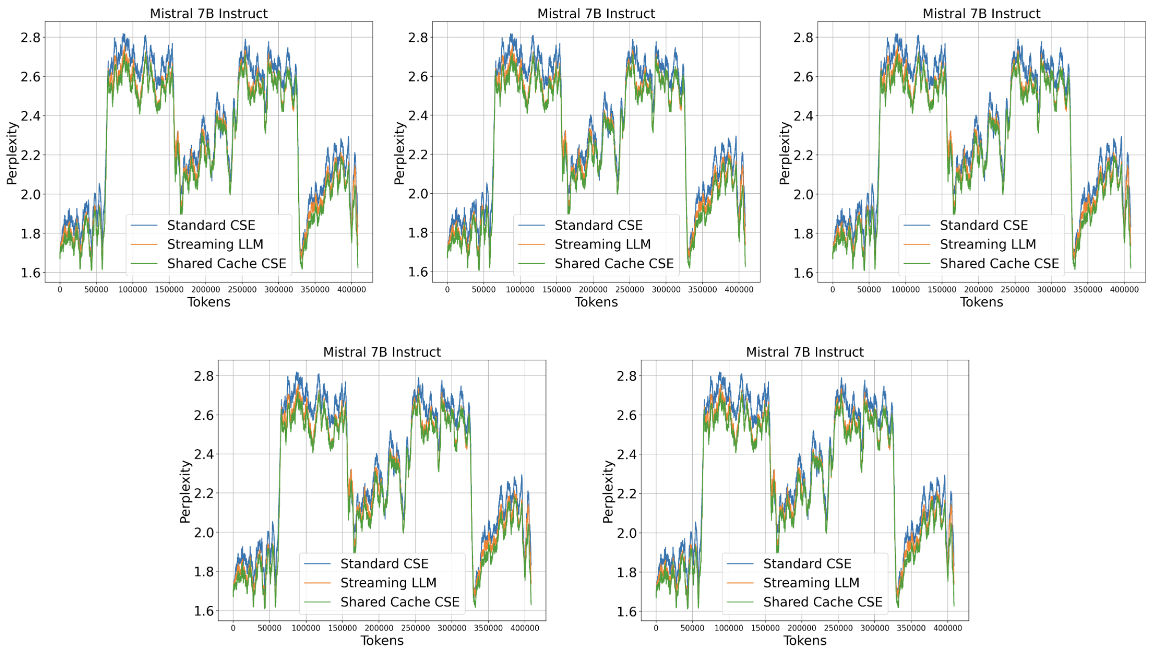

Figure 12: The language modeling results on the Mistral 7B Instruct model. The instructions 6 to 10 listed in table 12 are used for the Shared Cache CSE method, respectively. The line chart is smoothed with a window size of 4096 for better visibility.