Question Tells You Where the Answer Is: Intention-aware Long-Context KV Cache Compression

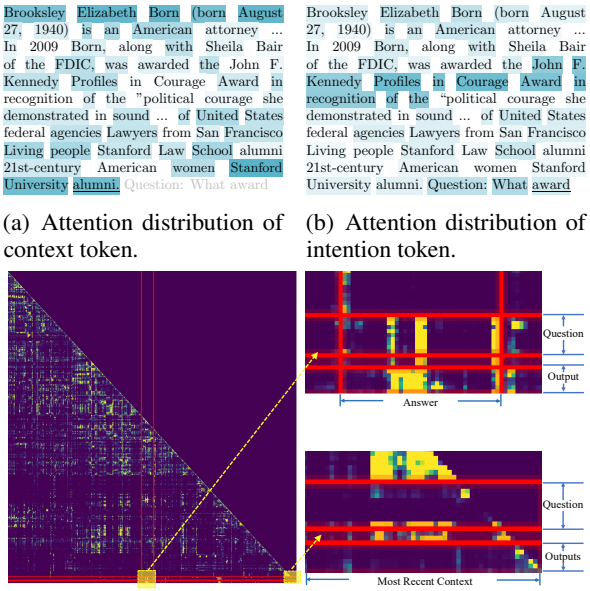
Liang Zhao[1], Xiaocheng Feng[1,2]*, Weihong Zhong[1], Lei Huang[1], Kun Zhu[1], Baoxin Wang[3], Dayong Wu[3], Guoping Hu[3], Ting Liu[1], Bing Qin[1,2]
[1]Harbin Institute of Technology [2]Peng Cheng Laboratory [3]iFLYTEK Research,
{lzhao, xcfeng, whzhong}@ir.hit.edu.cn

Abstract

The increasing context window greatly extends the capabilities of large language models, but on the other hand, it incurs an unaffordable memory overhead and computational latency due to the increasing Key-Value (KV) cache size. Recent KV cache compression methods manage to reduce the cache size by dropping irrelevant KVs. However, these methods often fail to identify crucial KVs for generation while excluding others accurately, resulting in severe information loss. To address this gap, we propose **IntentKV**, an intention-aware KV cache eviction method that identifies and retains crucial KVs according to the attention distribution of intention, which semantically reflects the user's goal and determines which part of the context is relevant. The consistency between the semantics and attention distribution is further substantiated through meticulously designed experiments. On this basis, IntentKV first distinguishes intention tokens from the vanilla context tokens based on their attention distribution distances. Then, the block-wise cumulative attention is calculated via aggregating the intention token attention. Finally, blocks that acquire high cumulative attention are picked and stored in KV cache. We evaluate our method across diverse long-context tasks and models. Results demonstrate that IntentKV can effectively maintain the model performance while reducing the KV cache size from 128K to 2K, leading to a 6.3x increase in decoding speed and 7.8x enhancement in memory efficiency compared to the default setting[1].

1 Introduction

Large language models (LLMs), ever since their birth, have drastically transformed numerous domains with their extraordinary capabilities (OpenAI, 2023; GLM et al., 2024; Dubey et al., 2024; Abdin et al., 2024; AI et al., 2024). Along with the growth of LLMs, their context windows have also been greatly extended, up to 1M tokens (Xiao et al., 2024). Such a long context on the one hand unlocks various applications of LLMs, but on the other hand, also poses a formidable challenge for inference. Specifically, the Key-Value (KV) cache, designed for the avoidance of duplicate computation and speed boost, can take up an unaffordable GPU memory and still incurs significant latency when processing long contexts (Fu, 2024). To deal with this challenge, numerous methods have been proposed to reduce the size of KV cache. However, previous methods either retain critical tokens



(a) Attention distribution of context token.

(b) Attention distribution of intention token.

(c) Actual attention map exhibiting high similarity between the distribution of question tokens and generated tokens.

Figure 1: (a) and (b) provides an intuitive comparison between attention distributions of context token and intention token in the same context, where underlined token denotes the query token and darker colors denote higher attention. (c) illustrates this difference in actual attention map, where the upper right figure demonstrates question and generated token both place more attention on the answer segment and the lower right figure demonstrates that they both exhibit low recency bias.

*Corresponding Author
[1]https://github.com/linkezh/IntentKV

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27153–27169
July 2-7, 2026

based on trivial historical information without a perception of generation query specificity (Liu et al., 2023; Zhang et al., 2023; Yang et al., 2024; Cai. et al., 2024), potentially dropping vital KVs for future generation, or only roughly estimate and load possible important KVs for each generation query (Tang et al., 2024), resulting in higher I/O and computation cost. Those approaches inevitably lead to suboptimal performance.

Different from existing methods, we are aware that critical tokens for generation can be marginal at the pre-filling stage, but we still argue that they can be identified based solely on historical information. From the perspective of pragmatics, ***intention*** *expresses what information is sought to be obtained from the addressee* (Huddleston and Pullum, 2005) and determines relevant context. In a long context, these **intention words** usually dominate the output of a model, e.g., a question based on a novel. Thus, these tokens possess a high degree of directivity, exhibiting a strong semantic tendency to the context segment helpful for response (Sperber and Wilson, 1986). Notably, this tendency is reflected in attention distribution, as represented in Fig. 1 (a) and (b). As depicted in Fig. 1 (c), when LLaMa-2-7b-chat (Touvron et al., 2023) is processing a long question-answering example (details in Fig. 6), the attention distribution of question (i.e., intention) tokens and generated tokens exhibit a high degree of similarity, significantly differing from which of vanilla context tokens. Intriguingly but not surprisingly, they both place disproportionate attention on the 'answer' in the context, providing a novel avenue to identify crucial KVs.

Motivated by the observations above, we propose **IntentKV**, an intention-aware KV cache eviction policy that efficiently and effectively identifies intention tokens and greatly reduces the size of the KV cache while maintaining superior performance. With a careful design, IntentKV demonstrates both efficiency and effectiveness in long context inference without the need of any additional training. To conclude, we make the following contributions:

- We, to the best of our knowledge, are the first to emphasize the role of intention in long-context inference. Highly consistent with pragmatics, our experiment reveals two key features of attention pattern of intention: (1) a high tendency to crucial context for response generation and (2) a markedly distinguished attention distribution compared to vanilla context tokens.
- Integrating above observation with intrinsic characteristics of attention mechanism, we propose intention-aware KV cache eviction policy, IntentKV, that effectively identify intention from a long context and retain critical KVs for generation according to its attention distribution.
- Through a comprehensive evaluation across various LLMs and benchmarks, we demonstrate the superiority of IntentKV, which can reduce the size of KV cache from 100K+ to 2K without an appreciable performance loss. Further ablation studies corroborated the robustness of IntentKV.

2 Related Work

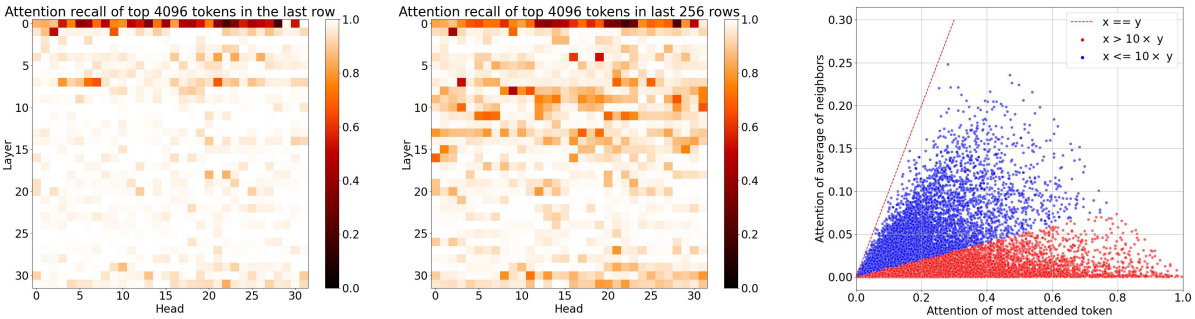
Long Context LLMs. Ever since the dawn of LLMs, longer context window has been a sought by both research community and industry, aiming to enable LLMs to answer questions on long text (Wang et al., 2024; Ma et al., 2025), summarize multi documents (Laban et al., 2024), resolve issues from real-world code repository (Jimenez et al., 2023), to name a few. As a result, a variety of methods have been developed to extend context window of LLMs (Wan et al., 2023; Zhao et al., 2024) and the maximum context length of LLMs has dramatically grown from few thousands (Brown et al., 2020) to 128K (OpenAI, 2023; Abdin et al., 2024; Dubey et al., 2024) and even more (GLM et al., 2024; Team, 2024a). However, this extraordinary journey towards longer context window also poses a greater challenge in the distraction (Shi et al., 2023) and latency (Fu, 2024) during inference phase.

KV Cache Compression and Eviction. Prior approaches compress context into "gist" tokens (Mu et al., 2023; Chevalier et al., 2023) or apply KV eviction based on attention patterns. Recent query-aware methods like SnapKV (Li et al., 2024) and Quest (Tang et al., 2024) improve over static policies but treat all context tokens equally, missing the semantic distinction between intention and vanilla context tokens (full survey in Appendix A).

IntentKV addresses this by explicitly identifying intention tokens via their distinct attention patterns using Jensen-Shannon Divergence, enabling more accurate one-time eviction with minimal overhead.

3 Attention Characteristics

In this section, we present a brief analysis of prominent characteristics of attention, which acts as a



(a) Attention is sparse.

(b) Specific sparsity varies with generation steps.

(c) Attention of tokens near the most attended one tends to be low.

Figure 2: (a) How much attention can top 4k tokens cover in a 124k context for one generation step. (b) If we consider multiple generation steps in a same example, the sparsity is largely weakened. (c) Attention of the most attended token and average attention of its neighbors in 19230 generations, where the input consist of 20 samples from every dataset in LongBench-E (Bai et al., 2024).

solid foundation for future discussion as well as our method, IntentKV.

3.1 Attention is Sparse

Sparsity, as an inductive bias inherent in the attention mechanism of pre-trained language models, has been a long-standing topic of research and has been extensively studied since the inception of these models, both empirically (Beltagy et al., 2020; Zaheer et al., 2020; Zhang et al., 2023; Jiang et al., 2024) and theoretically (Edelman et al., 2022; Likhosherstov et al., 2023). To intuitively show the sparsity of attention in long context, we take an example with 124K tokens from InfiniteBench (Zhang et al., 2024) and visualize the attention coverage of the top 4k tokens when generating the first new token (i.e., the last row of the $124k \times 124k$ attention matrix). As represented in Fig. 2(a), only the top 4k tokens gathered most attention, achieving an average recall of 96.6% across layers and heads.

This sparsity in the attention pattern of long context sequences motivates the research of KV cache eviction policies: we can expect negligible performance loss when evicting a large portion of KVs if the generation only relies on a small portion.

3.2 Attention is Dynamic

Although sparsity is an inherent property of the attention matrix and remains consistent across samples and generation steps, the specific distribution of attention is highly dependent on the input prompt and the token being generated (Likhosherstov et al., 2023). To illustrate how the model's focus shifts across steps, we use the attention matrix from Sec. 3.1 to compute attention recall. Unlike before, we sum the last 256 rows to capture each token's total attention over these generations. As presented in Fig. 2(b), the attention recall decreases significantly after aggregation, implying the attention distribution highly depends on the current generating token. A more concrete example is presented in Appendix B.1, which exhibits how attention focus consistently moves with generation steps. Since the specific attention sparsity pattern varies with query token, existing methods based solely on historical attention inevitably evict vital KVs for later generations, leading to suboptimal performance.

3.3 Attention is Scattered

In addition to the well-established dynamic sparsity of the attention matrix, we further observe that attention distribution is highly scattered. Specifically, rather than allocating substantial attention across consecutive tokens within the answer segment, the model tends to focus attention on a single token or few discrete tokens, while allocating much less attention to the neighboring tokens. In Figure 2(c), we present the attention of the most attended context tokens (excluding 32 initial and final tokens to reduce the impact of attention sink and recency bias) and the average attention directed towards its neighboring tokens, where the attention is taken from Llama-3.1-8B-Instruct (Dubey et al., 2024). Apparently, a significant portion of the most attended tokens are accompanied by tokens with significantly less attention, especially for those themselves acquiring high attention. These samples are more important since high attention on context tokens implies that the model is assertive about the relevance of the context tokens when generating semantic tokens, as in the case presented in Appendix B.1. In summary, this scatter pattern implies that instead of retaining top-k tokens as

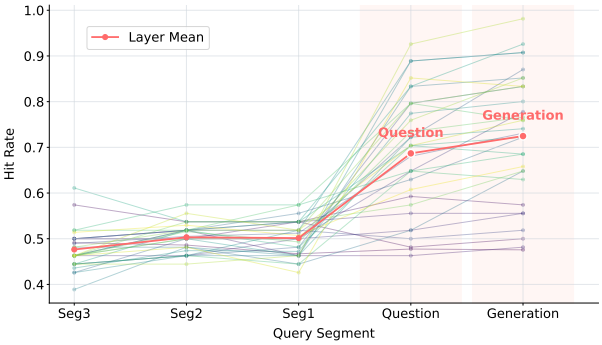


Figure 3: Hit rate of context selected by attention distribution of different query segments, where each line represents a different layer.

adopted in most previous studies (Liu et al., 2023; Zhang et al., 2023), a more appropriate approach is to retain KVs in block level, to avoid severe fragmentation as well as eviction of important KVs near the most attended tokens.

4 Intention

4.1 Attention Tendency of Intention

In pragmatics, intention refers to the goal of the speaker and denotes what information is sought to be obtained from the addressee, exhibiting strong semantic relevancy to the context segment helpful for response (Sperber and Wilson, 1986). In long-context inference, intention is usually expressed by several sentences (e.g., question in long-document question answering), dominating the generation and showcasing a strong attention tendency to essential context for generation, even when the model gives a wrong answer (Peysakhovich and Lerer, 2023). To quantitatively demonstrate this tendency, we calculate the extractive answer hit rate solely by attention distribution from different query segments of tokens (generation, question, and three preceding control segments), i.e., we pick the context gaining the highest attention from a segment and see if the answer is contained in it. As represented in Fig. 3, the hit rate of question tokens is very close to that of generated tokens and significantly higher than previous control segments in most layers. The average hit rate of question tokens and generated tokens seem not sufficiently close to one, as a result of different attention patterns across heads (Ge et al., 2023; Wu et al., 2024). Thus, we present setting details and a fine-grained analysis in Appendix C.2. Both results support that **the intention does tell us where the crux is**.

4.2 Distinguished Attention Distribution of Intention

As the intention puts disproportionate attention on the answer while context tokens exhibit strong attention sink effect and recency bias, it is a straightforward deduction that the attention distribution of intention differ significantly from previous vanilla context tokens. To illustrate this difference, we provide a concrete example in Appendix B.2. Note that we hypothesize the intention is always positioned in the last of the input in this paper so that it "has" attention distribution over whole context and the problem degenerates into a search for the boundary between vanilla context and intention instead of finding "a needle in a haystack", which greatly simplifies the problem without compromising the generality of our method.

5 IntentKV

5.1 Preliminaries and Problem Formulation

Typically, the inference of LLMs can be divided into two phases. In the *pre-fill* stage, the model processes all input tokens to compute intermediate states and stores them in the cache (i.e., KV cache), which are used to generate the first new token. In the following *decode* stage, the model generates output tokens one at a time in an autoregressive manner, until a stopping criterion is met. Formally, given query, key and value matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$, the pre-fill phase of attention is:

$$\text{Cache}_0 \longleftarrow \text{Save}(\mathbf{K}, \mathbf{V}, \mathbf{s}), \quad (1)$$

$$\mathbf{A} = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right), \quad \mathbf{O} = \mathbf{A}\mathbf{V}, \quad (2)$$

where $\mathbf{s} \in \{0, 1\}^n$ is a selection mask. Specifically, for any $t \in \mathbb{Z}, 1 \leq t \leq n$, $\mathbf{K}_t, \mathbf{V}_t$ will be stored into KV cache if and only if $\mathbf{s}[t] = 1$. The default $\mathbf{s}$ is $\mathbf{1}$, i.e., all KVs are stored. Then, without loss of generality, when the model is generating i-th token, where the inputs are $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i \in \mathbb{R}^d$, the decode phase of attention can be formulated as:

$$\text{Cache}_i, \mathbf{K}, \mathbf{V} = \text{Update}(\text{Cache}_{i-1}, \mathbf{k}_i, \mathbf{v}_i, \mathbf{s}_i)$$

$$\mathbf{a}_i = \text{Softmax}\left(\frac{\mathbf{q}_i\mathbf{K}^T}{\sqrt{d}}\right), \quad \mathbf{o}_i = \mathbf{a}_i\mathbf{V}.$$

We investigate whether we can reduce the KV cache *before* decode phase, i.e., instead of storing all keys and values in the pre-filling phase, we aim to find a policy $\mathcal{C}$ that returns a selection mask

Algorithm 1 IntentKV

Input: $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{n \times d}$, α, β, S

 # Calculate attention distributions of last α queries
$\mathbf{A}^{\alpha} = \text{Softmax}\left(\frac{\mathbf{Q}_{[-\alpha:]}\mathbf{K}}{\sqrt{d}}\right)$

 # Sum-pooling to smooth special tokens
$\mathbf{A}^{\alpha}_{pooled}$ = SumPool($\mathbf{A}$)

 # Calculate JSD
$\mathbf{d}_{JS} = \sqrt{JSD(\mathbf{A}^{\alpha}_{pooled} || \mathbf{A}^{\alpha}_{pooled_0})}$

 # Identify intention with highest differential JSD
intentIdx = argtop(diff($\mathbf{d}_{JS}$))

 # Calculate cumulative attention
$\mathbf{a}^{intent}$ = sum($\mathbf{A}^{\alpha}_{[\text{intentIdx}:]}$)

 # Divide cumulative attention into block sum
$\mathbf{a}^{block} = \{a_0, a_1, \ldots, a_{b-1}\}$, where $b = \lceil n/\beta \rceil$
for block index i from 0 to $b-1$ **do**
  $a_i \leftarrow sum(\mathbf{a}^{intent}_{[i\times\beta:\min((i+1)\times\beta, n)]})$
end for

 # Pick blocks with highest attention
blkIndices = argtop($\mathbf{a}^{block}$, $\lfloor S/\beta \rfloor$)

 # Construct selection mask
$\mathbf{s} = \mathbf{0}^n$
for block index i in blkIndices **do**
  for key index $j \in [(i-1) * \beta, \min(i * \beta, n)]$ **do**
    $\mathbf{s}_j \leftarrow 1$
  end for
end for
return $\mathbf{s}$

based on attention matrix $\mathbf{A}$, achieving a balance between memory/computation efficiency and attention expressiveness. This can be formally expressed as a multi-objective optimization problem:

$$\begin{aligned} &\min \sum |\mathcal{L}_{\mathcal{C}} - \mathcal{L}| \\ &\min |\text{Cache}'_0|, \\ &\quad \text{where Cache}'_0 \leftarrow \text{Save}(\mathbf{K}, \mathbf{V}, \mathcal{C}(\mathbf{A})) \end{aligned} \tag{3}$$

where $\mathcal{L}_{\mathcal{C}}$ is the prediction loss conditioning on $\mathcal{C}$ and $\mathcal{L}$ is the prediction loss without KV cache eviction. In a nutshell, the first objective aims to reduce performance drop after evictions based on $\mathcal{C}$, while the second one seeks to minimize the size of KV cache and thus memory and computation overhead. This objective is different from existing methods that aim to minimize the attention differences(Ge et al., 2023; Li et al., 2024), which will lead to suboptimal performance, considering the dispersed attention caused by long context (Chi et al., 2023). Thus, we are willing to retain only relevant context and exclude others, to help the model focus on important information and maintain performance.

5.2 Intention-aware KV Cache Eviction

To summarize observations presented in Section 4, we find (1) intention shows a strong attention tendency to the context that is crucial for later generation, which provides the possibility of retaining essential KVs before generation, and (2) this attention tendency leads to the distinguished attention distribution of intention, so we can leverage JSD to effectively and efficiently identify it. On this basis, we present IntentKV, an intention-aware KV cache policy mainly composed of two steps.

Intention identification based on JSD. Given that intention tokens exhibit a significantly different attention distribution from vanilla context tokens, we propose a simple yet effective method, using the square root of Jensen–Shannon divergence (JSD) to distinguish context tokens and intention tokens. For two attention distributions $\mathbf{a}_i, \mathbf{a}_j \in \mathbb{R}^n$, the square root of JSD can be calculated by:

$$\begin{aligned} D_{JS}(\mathbf{a}_i, \mathbf{a}_j) &= \sqrt{JSD(\mathbf{a}_i \parallel \mathbf{a}_j)} \\ &= \sqrt{\frac{1}{2}\left(D_{KL}(\mathbf{a}_i \parallel \mathbf{m}) + D_{KL}(\mathbf{a}_j \parallel \mathbf{m})\right)}, \end{aligned} \tag{4}$$

where $\mathbf{m} = \frac{1}{2}(\mathbf{a}_i + \mathbf{a}_j)$ and D_{KL} denotes the Kullback-Leibler divergence.

Formally, given an input $\mathbf{X} \in \mathbb{R}^{n \times d}$, we aim to find an index $l \in \{0, 1, \ldots, n\}$ that separates vanilla context and intention, i.e., $\mathbf{X}_{[0:l]}$ is the context and $\mathbf{X}_{[l:n]}$ is the intention. To this end, we select α queries $\mathbf{Q}^{\alpha} = \mathbf{Q}_{[-\alpha:]}$ from the end of the input, where α is a hyperparameter that should be greater than $n - l$ so that intention is fully included in, similar to observation window in previous works (Li et al., 2024; Cai. et al., 2024).

Then, we calculate the attention $\mathbf{A}^{\alpha}$ between $\mathbf{Q}^{\alpha}$ and $\mathbf{K}$ in order to acquire the attention distribution $\mathbf{a}^{\alpha}_i$ of each $\mathbf{q}^{\alpha}_i \in \mathbf{Q}^{\alpha}$. To smooth the noise incurred by special tokens such as <0x0A> and "." that also exhibit exceptional attention distributions, we use sum-pooling to aggregate information in the query and its s successors, i.e., $\bar{\mathbf{a}}^{\alpha}_i = \text{SumPool}(\mathbf{A}^{\alpha}_{[i:i+s]})$. Thereafter, JSD between subsequent pooled distributions and the first one (which is supposed to be vanilla context token) is calculated, and the position where the maximum JSD difference between consecutive tokens occurs is intention index l and all following tokens are deemed as intention tokens:

$$\begin{aligned} l &= \text{argtop}(\text{diff}(\mathbf{j})) - \alpha + n, \\ \text{where } \mathbf{j} &= \sqrt{JSD(\bar{\mathbf{A}}^{\alpha}, \bar{\mathbf{a}}^{\alpha}_0)}. \end{aligned}$$

Table 1: Performance (%) of different base models integrated with different KV eviction policies on Ruler(Hsieh et al., 2024), evaluated on context length from 4k to 128k. The capacity of KV cache is set to 4K for all KV cache eviction methods and context performance over 85.6% (performance of Llama-2-7B (Touvron et al., 2023) at 4K) is considered effective according to original setting.

Models	Methods	Eff.	4K	8K	16K	32K	64K	128K	Avg.
Llama3.1 (128K)	FullKV	32K	95.2	92.8	93.7	87.6	85.3	78.0	88.8
	StreamingLLM	4K	95.2	54.3	39.3	20.5	17.4	11.5	39.7
	SnapKV	16K	95.2	88.4	87.3	73.5	64.3	52.8	76.9
	IntentKV	**32K**	95.2	**92.7**	**90.9**	**85.7**	**81.9**	**75.4**	**87.0**
Llama3 (262K)	FullKV	16K	92.3	87.5	87.5	83.5	82.5	78.7	85.3
	StreamingLLM	4K	92.3	50.0	34.5	17.7	15.1	11.8	36.9
	SnapKV	4K	92.3	78.8	74.9	65.5	60.0	53.6	70.9
	IntentKV	**16K**	92.3	**87.5**	**85.7**	**83.9**	**78.9**	**75.9**	**84.0**
Qwen (128K)	FullKV	32K	94.6	91.6	90.1	87.2	68.9	25.9	76.4
	StreamingLLM	4K	94.6	56.0	41.1	22.4	14.5	9.1	39.6
	SnapKV	4K	94.6	79.6	68.5	57.2	32.3	12.1	57.4
	IntentKV	**16K**	94.6	**91.9**	**89.6**	**85.1**	**64.5**	**29.2**	**75.8**
Yi (200K)	FullKV	8K	92.8	89.7	81.3	72.1	66.9	63.7	77.7
	StreamingLLM	4K	92.8	52.5	33.8	17.2	14.9	12.1	37.2
	SnapKV	4K	92.8	76.3	56.1	40.1	27.4	24.4	52.8
	IntentKV	**8K**	92.8	**88.6**	**79.7**	**70.8**	**67.4**	**60.4**	**76.6**

KV reservation based on cumulative attention. Once intention delimiter l has been identified, we calculate cumulative attention distribution of them, where $\mathbf{a}^{Intent} = \sum_{i=l}^{n}(\mathbf{A}[i,:])$, following existing methods (Zhang et al., 2023; Ge et al., 2023). However, instead of directly picking KVs gaining the most attention, we retain KVs at block level with block size β, considering the scattering characteristic of attention described in Section 3.3. After the final attention score of each block is calculated, we select blocks with the highest scores and only store corresponding KVs in KV cache, which can be formally described as $\mathbf{I}^{block} = \text{argtopk}(\mathbf{a}^{block}, k)$ where $\mathbf{a}^{block} = [a_0^{block}, a_1^{block}, \ldots, a_{n/\beta}^{block}]$ and each $a_i^{block} = \sum_{k=i\beta}^{\min((i+1)\beta,n)} a_k^{Intent}$. The complete procedure of IntentKV is presented in Algorithm 1, which takes the query matrix $\mathbf{Q}$, key matrix $\mathbf{K}$, initial observation window size α, block size β and size of KV cache S as input.

6 Experiments

In this section, we conduct a series of experiments to thoroughly evaluate the effectiveness and efficiency of IntentKV, covering a variety of long-context LLMs and datasets. Additional results on LongBench and needle-in-a-haystack test (gkamradt, 2024) is presented in Appendix F.

6.1 Setting

Models. We leverage four trending long context LLMs in consideration of their reputation for dealing with long sequences: (i) Meta-Llama-3.1-8B-Instruct-128k (Dubey et al., 2024) (**Llama3.1**) (ii) LLaMA3-8B-Instruct-262k[2] (**Llama3**) (iii) Yi-9B-200K (AI et al., 2024) (**Yi**) (iv) Qwen2.5-7B-Instruct-128k (Team, 2024b) (**Qwen**), where Yi is a pre-trained model and all others are instruction fine-tuned models. We use the default chat template for all instruction fine-tuned models in the subsequent experiments.

Datasets. We evaluate IntentKV on two challenging benchmarks consisting of long samples: (i) **InfiniteBench** (Zhang et al., 2024): This benchmark is tailored for evaluating the capabilities of LLMs to process, understand, and reason over super long contexts with 100K+ tokens. Our experiment utilizes 10 tasks of this benchmark, covering retrieval tasks such as Number retrieval and key-value retrieval, as well as realistic tasks including question-answering, coding, and summarization. (ii) **RULER** (Hsieh et al., 2024): This benchmark extends the vanilla NIAH test to enable effective evaluation of real context window of long-context LLMs with configurable sequence lengths. It consists of 4 categories (in total 13 tasks), including retrieval, multi-hop tracing, and QA tasks.

Baselines. We evaluate IntentKV against three strong fine-tuning-free baselines: 1) **FullKV**, where all KVs are stored into KV cache and can be utilized for later generation, is the default genera-

[2] https://huggingface.co/gradientai/Llama-3-8B-Instruct-262k

Table 2: Performance of KV eviction policies with different base models on InfiniteBench (Zhang et al., 2024), where the **bold** results indicate the highest value in each column excluding FullKV row.

	Methods	En.Sum	En.QA	En.MC	En.Dia	Zh.QA	Code.Debug	Math.Find	Retr.PassKey	Retr.Num	Retr.KV	Avg.
Llama3.1	FullKV	32.6	27.2	69.0	16.5	35.36	15.0	24.9	99.7	99.7	53.8	47.4
	StreamingLLM	20.6	19.6	69.0	7.5	25.2	**16.0**	24.9	3.7	3.2	0.6	19.0
	SnapKV	**27.9**	22.0	69.0	13.5	32.3	15.5	24.9	99.7	94.1	0.0	39.9
	IntentKV	26.5	**25.1**	**69.0**	**14.0**	**34.3**	15.5	**25.1**	**99.7**	**97.8**	**50.6**	**45.8**
Llama3	FullKV	18.1	12.3	63.8	0.5	20.3	24.6	18.0	100.0	100.0	6.8	36.4
	StreamingLLM	14.0	8.0	63.8	0.5	16.6	24.6	18.0	3.4	3.4	0.8	15.3
	SnapKV	**15.4**	11.9	63.8	0.5	19.2	24.6	18.0	100.0	77.8	0.2	33.1
	IntentKV	15.1	**12.6**	**63.8**	**0.5**	**20.3**	**24.6**	**18.0**	**100.0**	**100.0**	**10.0**	**36.5**
Qwen	FullKV	18.8	5.0	45.0	12.0	9.2	5.9	38.0	100.0	92.4	0.0	32.6
	StreamingLLM	**20.6**	3.9	45.0	5.0	8.3	5.8	31.4	3.4	1.7	**0.2**	12.5
	SnapKV	18.0	4.8	45.0	8.0	10.0	5.8	34.3	94.2	28.8	0.0	24.9
	IntentKV	17.6	**5.0**	**45.0**	**9.5**	**10.3**	**5.8**	**36.6**	**100.0**	**97.8**	0.0	**32.8**
Yi	FullKV	8.3	10.9	64.2	2.0	17.3	20.0	23.4	100.0	99.7	27.6	37.3
	StreamingLLM	**11.2**	6.9	**67.3**	2.0	13.6	**20.8**	**25.7**	3.4	3.4	1.2	27.0
	SnapKV	4.6	10.9	64.2	1.5	16.3	18.6	23.7	96.3	**15.4**	0.0	25.1
	IntentKV	4.8	**10.9**	66.4	**2.0**	**16.5**	18.6	23.4	**100.0**	**99.5**	**23.2**	**36.5**

tion setting for current LLMs; 2) **StreamingLLM**, where only initial few KVs and most recent KVs are retained in the KV cache, is an effective extension of traditional sliding window mechanism; 3) **SnapKV**, where the critical KVs are selected by attention distribution of a fixed observation window in the last of the input individually, shares most similarities with our method. Methods such as PyramidKV, CAKE, and Ada-KV are excluded from our main experiments because they are **conceptually orthogonal** to our approach. While IntentKV and the selected baselines focus on accurately identifying salient KV pairs, these other methods address coarse-grained cache budget allocation, such as on a per-layer or per-head basis, by leveraging structural imbalances in the attention mechanism. However, we present the performance comparison on LongBench between IntentKV as well as PyramidKV and CAKE in appendix E.1.

Implementation Details. For all KV eviction methods, we set the capacity of KV cache to 2048 for InfiniteBench and 4096 for Ruler. To ensure fairness, the observation window of SnapKV as well as the initial size of queries α in IntentKV are both 64. The default block size of IntentKV is set to 16. For the key-value retrieval task in InfiniteBench (Retr.KV), we increase the block size to 128, because each retrievable key-value pair spans approximately 50 tokens and token-level eviction would fragment these pairs; block-level eviction at 128 ensures each pair is retained atomically. For StreamingLLM, the first 128 tokens are always retained and the remaining capacity stores the most recent KVs. Details on FlashAttention compatibility are provided in Appendix C.1.

6.2 Main Results

Ruler. As presented in Table 1, our approach demonstrates significant advantages, with performance that substantially outperforms StreamingLLM and SnapKV on Ruler. Specifically, IntentKV can consistently maintain performance across all lengths of context. At the same time, other eviction methods cause severe performance loss once context length is beyond the capacity of KV cache, implying that they drop crucial KVs for response while IntentKV did not. Even compared with FullKV, IntentKV incurs a negligible accuracy loss and demonstrates some degree of superiority in certain cases, supporting that IntentKV can help the model focus on crucial information. Considering the variety of tasks and difficulty in Ruler, we deem this as strong support for the special status of intention in long context as well as the effectiveness of our method.

InfiniteBench. As the Table 2 shows, IntentKV demonstrates state-of-the-art overall performance compared to previous methods. Further, even reducing the size of KVs from 100K+ to 2K, our method leads to a negligible performance drop compared with the FullKV setting, which is a powerful demonstration of the superiority of our method. Regarding the performance on specific tasks, we note that for some simple tasks where intention is relatively long compared to our initial window 64 (e.g., En.MC), both SnapKV and IntentKV can achieve comparable performance with FullKV, while for more challenging tasks where intention posses stronger directivity such as QA, IntentKV demonstrates significant superiority. The striking gap in Retr.KV between SnapKV and In-

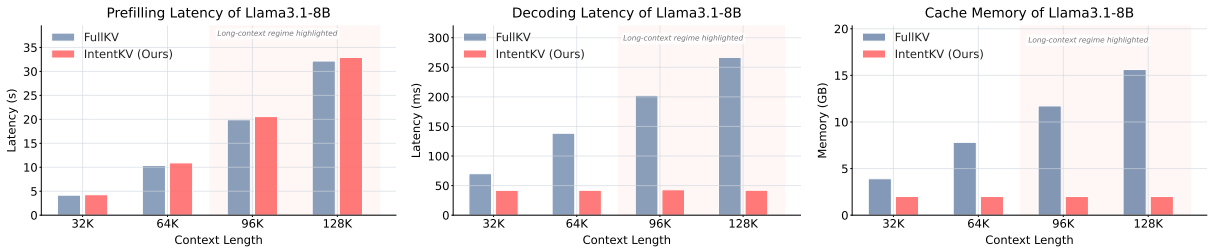


Figure 4: Efficiency evaluation of IntentKV with Llama3.1-8B compared to FullKV.

tentKV comes from our block selection strategy. Due to the scattered characteristic described in Section 3.3, it is challenging to extract a complete key-value pair (about 50 tokens) at the token level. However, we do observe one limitation of IntentKV in this experiment. When dealing with tasks such as summarization where the intention is extremely short and possesses little directivity, IntentKV exhibits degraded performance. We leave the extension of IntentKV to these tasks for future work.

Efficiency. Effectiveness and efficiency are two core concerns of KV cache eviction methods, and the former has been thoroughly discussed in previous sections. Hence, we present a comparative analysis in this part to demonstrate the superiority of IntentKV in efficiency.

As presented in Fig. 4, with a fixed capacity of KV cache as 2048 (which can maintain most performance as previous results suggested), IntentKV can ensure a constant decoding speed and memory overhead of KV cache regardless of the input length, while FullKV setting incurs a linear increase in both decoding latency and memory footprint as input length escalates. When the input length comes to 128K, IntentKV can reduce the decoding latency to 16% and the memory overhead of KV cache to 13% compared to FullKV. Further, IntentKV leads to similar pre-filling latency compared to FullKV, which means it incurs little to no additional cost to integrate IntentKV in long context inference. Considering Llama3-8B is based on grouped-query attention (GQA) (Ainslie et al., 2023), we also conduct efficiency evaluation on Llama2-7B in Appendix E.2.

Table 3: Performance (%) of Llama3.1 integrated with IntentKV of different block sizes on Ruler(Hsieh et al., 2024).

Block Size	Eff.	4K	8K	16K	32K	64K	128K	Avg.
1	8K	95.2	85.9	74.6	59.7	51.0	36.0	67.1
4	16K	95.2	91.2	87.2	77.7	68.7	58.2	79.7
8	16K	95.2	92.4	90.7	83.1	73.7	64.4	83.3
16	32K	95.2	92.7	90.9	85.7	81.9	75.4	87.0
32	32K	95.2	92.5	92.2	87.0	81.3	73.9	87.0
64	32K	95.2	92.7	91.8	85.9	81.7	74.5	87.0
128	32K	95.2	92.7	90.9	85.7	81.9	75.4	87.0
256	32K	95.2	92.3	91.9	86.0	82.8	75.6	87.3

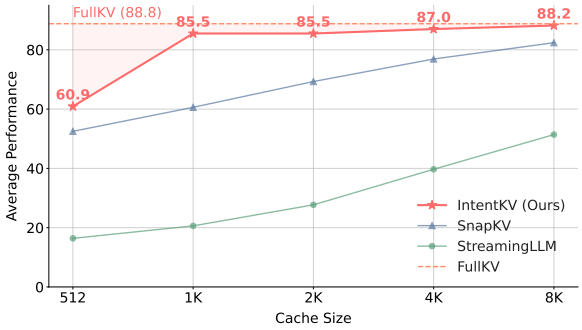


Figure 5: Average performance of LLama3.1 on Ruler with KV eviction methods of various cache sizes.

6.3 Ablation on Block Size

The block size is a vital hyperparameter we introduced, which controls the information granularity we retain. Here, we analyze the impact of block size on IntentKV. From Table 3, we can draw two conclusions, (1) the introduction of block is essential to IntentKV to avoid semantic fragmentation and eviction of crucial KVs, as removal of block mechanism (block size 1) and small block sizes lead to severe performance degradation; (2) IntentKV demonstrates a low sensitivity in block size, as performance remains stable once block size goes beyond 16. The complete ablation on block mechniasm is in Appendix D.2.

6.4 Ablation on KV Cache Size

Considering that the size of KV cache determines the volume of information to be reserved and thus has a significant impact on KV eviction methods as well as model performance, we report the performance of Llama3.1 on Ruler coupled with different cache sizes under different eviction methods. As presented in Fig. 5, despite clearly benefiting from larger size of KV cache, IntentKV can preserve a relatively stable performance even when the size of KV cache is minimal, e.g., 1% of the context. Compared to other methods, IntentKV demonstrates

Table 4: Performance (%) of Llama3.1 integrated with KV eviction methods on Ruler (Hsieh et al., 2024), evaluated across context lengths from 4K to 128K under varying KV cache capacities. Context performance above 85.6% (the score of Llama-2-7B (Touvron et al., 2023) at 4K) is considered effective per the original benchmark setting.

Method	Cache Size	Eff.	4K	8K	16K	32K	64K	128K	Avg.
IntentKV	512	<4K	69.1	65.1	66.8	61.5	55.7	47.1	60.9
	1024	4K	91.6	84.7	82.6	80.7	76.7	66.7	80.5
	2048	16K	95.0	89.9	89.3	84.3	81.6	72.6	85.5
	4096	32K	95.2	92.7	90.9	85.7	81.9	75.4	87.0
	8192	32K	95.2	92.8	93.3	86.6	83.7	77.4	88.2
SnapKV	512	<4K	68.0	59.8	59.6	47.0	45.3	35.2	52.5
	1024	<4K	79.7	69.4	68.2	53.2	52.1	41.1	60.6
	2048	4K	88.7	81.4	77.0	62.0	57.1	49.7	69.3
	4096	8K	95.2	88.4	87.3	73.5	64.3	52.8	76.9
	8192	16K	95.2	92.8	92.7	80.7	72.6	60.3	82.4
StreamingLLM	512	<4K	29.3	17.3	15.7	13.2	12.9	10.2	16.4
	1024	<4K	44.7	22.5	19.6	13.8	12.8	10.4	20.6
	2048	<4K	63.8	36.3	26.5	15.7	13.5	10.5	27.7
	4096	4K	95.2	54.3	39.3	20.5	17.4	11.5	39.7
	8192	8K	95.2	92.8	57.1	27.9	19.8	15.5	51.4

significant superiority in all cache size settings, especially when the size is small, which is a strong demonstration of the capability of IntentKV to identify and retain crucial information for generation. The full results of Llama3.1-8B integrated with different KV eviction methods of various KV cache sizes on Ruler is presented in Table 4. From this table, we can conclude that IntentKV outperforms other methods on all KV cache sizes, especially when the size of KV cache is limited, which is a strong endorsement of IntentKV's ability to better identify and retain content that is important for generation.

Additional ablation studies on IntentKV are presented in Appendix D.

7 Conclusion

In this paper, we leverage the notion of intention in pragmatics to reduce KV cache, which denotes the goal of the speaker and determines relevant context. As a solid foundation, we first demonstrate two attention features of intention, (1) intention tokens exhibit a high degree of similarity in attention distribution with generated tokens, both focusing on context crucial for generation; and (2) intention tokens differ significantly from vanilla context tokens in attention distribution. Combining these observations with intrinsic characteristics of the attention mechanism, we design a novel KV cache eviction policy IntentKV that can identify intention from a long context and evict KVs accordingly, which incurs minimal performance loss while significantly enhancing memory and speed efficiency.

Limitations

Despite the strengths, there are a few limitations in the current implementation of IntentKV. First, as stated in Section 4.2, we always hypothesize intention is located in the end of context in this paper. However, we argue this is a sound hypothesis. On the one hand, placing questions at the beginning or middle of a long context can degrade performance due to recency bias in language models (Peysakhovich and Lerer, 2023), which also explains why existing long benchmarks all place the questions at the end of the prompt (Bai et al., 2024; Zhang et al., 2024; Hsieh et al., 2024). On the other hand, even if the question is placed at the beginning of the context initially, it is always beneficial to repeat the question at the end of the context to enhance the understanding process(Xu et al., 2024; Liu et al., 2024). Another limitation of IntentKV is the marginal efficiency enhancement when the context is relatively short for GQA-based LLMs, as we discussed in Appendix E.2. Moreover, IntentKV, together with other KV compression and eviction methods, can not be easily extended to scenarios like multi-turn chat, as they aim to discard irrelevant KVs, and the culling is irreversible. Nonetheless, our contributions offer significant insights and tools for the community, providing a new angle for KV cache compression problem and stimulating a more refined method for efficient inference.

Acknowledgements

Xiaocheng Feng is the corresponding author of this work. We thank the anonymous reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China (NSFC) (grant 62522603, 62276078), the Key R&D Program of Heilongjiang via grant 2022ZX01A32, the Fundamental Research Funds for the Central Universities (XNJKKGYDJ2024013) .

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, and 65 others. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. *Preprint*, arXiv:2404.14219.

01 AI, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, and 12 others. 2024. Yi: Open Foundation Models by 01.AI. *Preprint*, arXiv:2403.04652.

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.

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. 2024. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *Preprint*, arXiv:2004.05150.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Zefan Cai., Yichi Zhang, Bofei Gao, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2024. PyramidKV: Dynamic KV Cache Compression based on Pyramidal Information Funneling. *Preprint*, arXiv:2406.02069.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and 1 others. 2024. Magicpig: Lsh sampling for efficient llm generation. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting Language Models to Compress Contexts. *Preprint*, arXiv:2305.14788.

Ta-Chung Chi, Ting-Han Fan, and Alexander I. Rudnicky. 2023. Attention Alignment and Flexible Positional Embeddings Improve Transformer Length Extrapolation. *Preprint*, arXiv:2311.00684.

Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *Preprint*, arXiv:2307.08691.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 514 others. 2024. The Llama 3 Herd of Models. *Preprint*, arXiv:2407.21783.

Benjamin L. Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. 2022. Inductive Biases and Variable Creation in Self-Attention Mechanisms. In *Proceedings of the 39th International Conference on Machine Learning*, pages 5793–5831. PMLR.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2024. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *arXiv preprint arXiv:2407.11550*.

Yao Fu. 2024. Challenges in Deploying Long-Context Transformers: A Theoretical Peak Performance Analysis. *Preprint*, arXiv:2405.08944.

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. In *The Twelfth International Conference on Learning Representations*.

gkamradt. 2024. Gkamradt/LLMTest_NeedleInAHaystack.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, and 39 others. 2024. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. *Preprint*, arXiv:2406.12793.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the Real Context Size of Your Long-Context Language Models? *Preprint*, arXiv:2404.06654.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Rodnry Huddleston and Geqffrry Pullum. 2005. The cambridge grammar of the english language. *Zeitschrift für Anglistik und Amerikanistik*, 53(2):193–194.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. MInference 1.0: Accelerating Pre-filling for Long-Context LLMs via Dynamic Sparse Attention. *Preprint*, arXiv:2407.02490.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2023. SWE-bench: Can Language Models Resolve Real-world Github Issues? In *The Twelfth International Conference on Learning Representations*.

Philippe Laban, Alexander Fabbri, Caiming Xiong, and Chien-Sheng Wu. 2024. Summary of a Haystack: A Challenge to Long-Context LLMs and RAG Systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9885–9903, Miami, Florida, USA. Association for Computational Linguistics.

Kunxi Li, Zhonghua Jiang, Zhouzhou Shen, Zhaode Wang, Chengfei Lv, Shengyu Zhang, Fan Wu, Fei Wu, and Wanxiang Che. 2025. MadaKV: Adaptive Modality-Perception KV Cache Eviction for Efficient Multimodal Long-Context Inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. SnapKV: LLM Knows What You are Looking for Before Generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Valerii Likhosherstov, Krzysztof Choromanski, and Adrian Weller. 2023. On the Expressive Flexibility of Self-Attention Matrices. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):8773–8781.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the Persistence of Importance Hypothesis for LLM KV Cache Compression at Test Time. *Advances in Neural Information Processing Systems*, 36:52342–52364.

Weitao Ma, Xiyuan Du, Xiaocheng Feng, Lei Huang, Yichong Huang, Huiyi Zhang, Xiaoliang Yang, Baohang Li, Xiachong Feng, Ting Liu, and 1 others. 2025. One for all: Update parameterized knowledge across multiple models with once edit. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16021–16034.

Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to Compress Prompts with Gist Tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352.

OpenAI. 2023. GPT-4 Technical Report. *Preprint*, arXiv:2303.08774.

Alexander Peysakhovich and Adam Lerer. 2023. Attention Sorting Combats Recency Bias In Long Context Language Models. *Preprint*, arXiv:2310.01427.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. 2025. Cake: Cascading and adaptive kv cache eviction with layer preferences. In *The Thirteenth International Conference on Learning Representations*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large Language Models Can Be Easily Distracted by Irrelevant Context. In *Proceedings of the 40th International Conference on Machine Learning*, pages 31210–31227. PMLR.

Dan Sperber and Deirdre Wilson. 1986. *Relevance: Communication and cognition*, volume 142. Harvard University Press Cambridge, MA.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference. *Preprint*, arXiv:2406.10774.

Gemini Team. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Qwen Team. 2024b. Qwen2.5: A party of foundation 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, and 49 others. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *Preprint*, arXiv:2307.09288.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. 2023. Efficient Large Language Models: A Survey. *Preprint*, arXiv:2312.03863.

Cunxiang Wang, Ruoxi Ning, Boqi Pan, Tonghui Wu, Qipeng Guo, Cheng Deng, Guangsheng Bao, Xiangkun Hu, Zheng Zhang, Qian Wang, and Yue Zhang. 2024. NovelQA: Benchmarking Question Answering on Documents Exceeding 200K Tokens. *Preprint*, arXiv:2403.12766.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval Head Mechanistically Explains Long-Context Factuality. *Preprint*, arXiv:2404.15574.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024. InfLLM: Training-Free Long-Context Extrapolation for LLMs with an Efficient Context Memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2025. DuoAttention: Efficient Long-Context LLM Inference with Retrieval and Streaming Heads. In *The Thirteenth International Conference on Learning Representations*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient Streaming Language Models with Attention Sinks. *Preprint*, arXiv:2309.17453.

Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, Jian-Guang Lou, and Shuai Ma. 2024. Re-reading improves reasoning in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15549–15575.

Dongjie Yang, XiaoDong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024. PyramidInfer: Pyramid KV Cache Compression for High-throughput LLM Inference. *Preprint*, arXiv:2405.12532.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024. ınftyBench: Extending Long Context Evaluation Beyond 100K Tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand. Association for Computational Linguistics.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H$_2$O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. *Preprint*, arXiv:2306.14048.

Liang Zhao, Xiachong Feng, Xiaocheng Feng, Weihong Zhong, Dongliang Xu, Qing Yang, Hongtao Liu, Bing Qin, and Ting Liu. 2024. Length Extrapolation of Transformers: A Survey from the Perspective of Positional Encoding. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9959–9977, Miami, Florida, USA. Association for Computational Linguistics.

Xiabin Zhou, Wenbin Wang, Minyan Zeng, Jiaxian Guo, Xuebo Liu, Li Shen, Min Zhang, and Liang Ding. 2024. Dynamickv: Task-aware adaptive kv cache compression for long context llms. *arXiv preprint arXiv:2412.14838*.

A Additional Related Work

KV cache eviction methods have evolved significantly. Early approaches like H_2O (Zhang et al., 2023) retain tokens with highest cumulative attention based on historical attention patterns. Subsequent work leveraged observations that initial tokens gather disproportionate attention (Xiao et al., 2023) and combined this with sliding window mechanisms.

Model-structure-aware methods emerged to exploit the heterogeneity of attention heads. Some work constructs adaptive KV caches based on attention patterns across different heads (Ge et al., 2023; Feng et al., 2024), while others exploit pyramidal information funneling across layers (Yang et al., 2024; Cai. et al., 2024). Recent approaches introduce task-aware layer budget allocation (Zhou et al., 2024) and jointly consider spatial and temporal dimensions (Qin et al., 2025).

Beyond query-aware methods, additional directions include leveraging locality-sensitive hashing for key sampling (Chen et al., 2024), retrieval head identification for head-wise budget allocation (Xiao et al., 2025), and extensions to multimodal settings (Li et al., 2025).

B Example and Case Study

B.1 Attention Focus Consistently Moves with Generation Steps

During experiments, we consistently observed that Layer 8 Head 25 placed the highest attention on the precedent token of the source token when the model is directly retrieving content from context, a similar but not the same behavior of retrieval head (Wu et al., 2024). To illustrate this point intuitively, we present an example here (note that this pattern is widely emerging across different samples from different datasets). The input is a sample from MultiFieldQA-en of LongBench, which is also the example in Fig. 1, as shown in Figure 6, and the attention scores on the answer segment of each generation step in Layer 8 Head 25 is represented in Fig. 7.

From this intriguing pattern, it is conjecturable that, 1) when the model is retrieving or copying a word from context, its attention peaks do not necessarily fall on the source token. Instead, some heads may place the attention peak on tokens near the source token with high semantic relevance to help locate the target source token and 2) the position of attention peak changes as generation progresses, which means even when inferencing on one sample, the generation of each token possibly relies on a very different part of the context, leading to the degraded performance of static sparse attention mechanisms (Beltagy et al., 2020; Zaheer et al., 2020) and KV cache policies that evict KVs based solely on historical information (Liu et al., 2023; Zhang et al., 2023; Ge et al., 2023).

Example from MultiFieldQA-en of LongBench

Question: What award did Brooksley Born receive in 2009?

Context: *...In 2009 Born, along with Sheila Bair of the FDIC, was awarded the John F. Kennedy Profiles in Courage Award in recognition of the "political courage she demonstrated in sounding early warnings about conditions that contributed" to the 2007–08 financial crisis. According to Caroline Kennedy, "Brooksley Born recognized that the financial security of all Americans was being put at risk by the greed, negligence and opposition of powerful and well connected interests"...*
Context Length: 3306 tokens

Prediction: `'The John F. Kennedy Profiles in Courage Award</s>'`

Figure 6: Main information of the example from MultiFieldQA-en of LongBench and the prediction of Llama2-7B-Chat. The segment containing the answer is marked in blue.

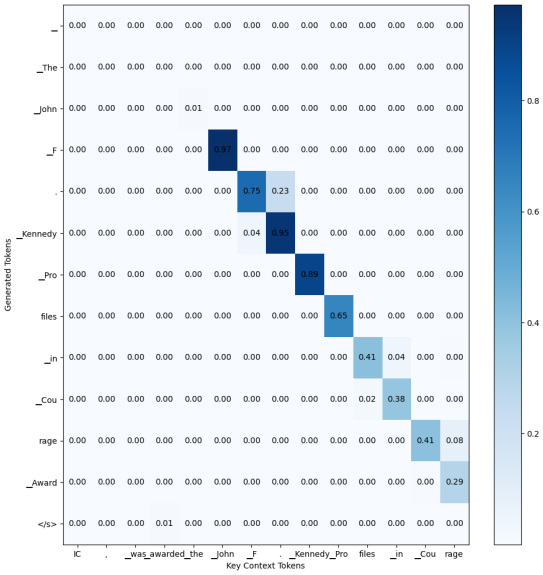


Figure 7: Attention score on the answer segment in Head 25 and Layer 8 of Llama2-7B-Chat when processing example in Figure 6. Every row represents the attention score when generating each answer token (ordered from top to bottom).

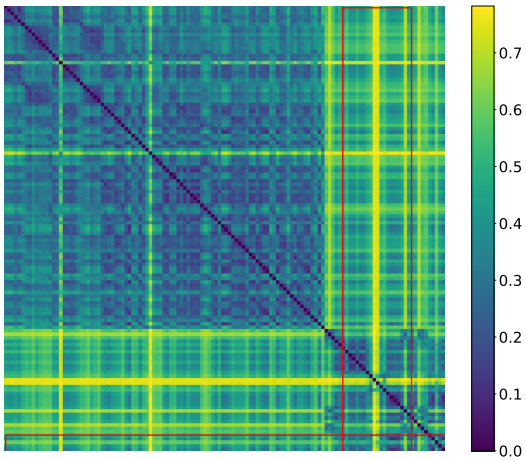


Figure 8: Jensen–Shannon divergence between attention distribution of last 128 tokens, where the vertical box denotes question segment and the horizontal box denotes generated segment.

B.2 Distances of Attention Distributions

To illustrate the claim that intention exhibits a distinguished attention pattern, we visualize the JSD between attention distributions of last 128 tokens when Llama-3.1-8B-Instruct (Dubey et al., 2024) is processing a long question-answering sample. In short, two observations can be drawn from Fig. 8, (1) except itself, generation segment exhibits the smallest JSD from question segment, supporting the argument presented in Section 4.1 again, and (2) question segment shows high JSD from all tokens before, supporting the feasibility of using JSD for intention identification.

C Details of Experiments

C.1 Experiment Environment

Our experiments are conducted in a computing environment equipped with 8 NVIDIA A100 GPUs with 80GB of memory in the bfloat16 format, while all of which can be done with a single A100. FlashAttention-2 (Dao, 2023) is leveraged to speed and scale inference. All experiments were conducted using greedy decoding to guarantee consistency across results.

To preserve FlashAttention compatibility, instead of extracting the full attention matrix from the pre-filling pass, we recompute attention over a small observation window of size α after pre-filling. This recomputation has $O(\alpha n)$ complexity, far below the $O(n^2)$ cost of the pre-filling attention, and introduces only marginal latency overhead as shown in Figure 4. All other components of the model—including the pre-filling and decoding phases—remain unmodified, making IntentKV a lightweight plug-in compatible with dynamic batching and prefix caching.

C.2 Hit Rate Experiment

Specifically, we consider special query segments including generation and question, as well as three control segments exactly before the question, each in size 16. Then, we divide the input into blocks of size 64 and select the top 10 blocks based on the aggregated attention distribution of each query segment, which leads to a hit if these blocks contain the answer. We sample three attention heads from every layer of Llama-3.1-8B-Instruct (Dubey et al., 2024) except for layers 0, 8, 16, 24, and 32, where all heads are taken to comprehensively investigate this phenomenon. We collect their attention distributions when processing 20 samples from HotpotQA-e of LongBench with an average length of 4466 tokens.

Table 5: Hit rate of query-sensitive heads and query-insensitive heads, where Q/G denote question/generation segment.

Group	Seg3	Seg2	Seg1	Q	G	Size
Sen.	44.7%	49.6%	49.8%	82.3%	88.7%	112
Ins.	50.4%	50.8%	49.2%	53.2%	54.0%	129
Tot.	47.8%	50.3%	49.5%	66.8%	70.1%	241

Considering some attention heads are insensitive to query (Ge et al., 2023; Jiang et al., 2024), we divide all heads into query-sensitive and query-insensitive groups by the hit rate of generation segment, i.e., if the hit rate of generation segment is low (e.g., less than 0.7), we deem this head as insensitive to query. As presented in Table 5, the hit rate of the question segment is even higher for these query-sensitive heads, further supporting that **the intention does tell us where the crux is**.

Table 6: Performance (%) of Llama3.1 integrated with IntentKV of different initial observation window sizes (α) on Ruler(Hsieh et al., 2024).

α	Eff.	4K	8K	16K	32K	64K	128K	Avg.
32	32K	95.2	92.1	92.0	87.0	78.5	71.4	86.0
64	32K	95.2	92.7	91.6	85.9	79.4	70.2	85.9
128	32K	95.2	92.3	90.7	86.5	79.8	69.9	85.7
256	16K	95.2	92.8	90.4	83.7	76.4	68.0	84.4

D Ablation Studies

D.1 Ablation on Observation Window

In this part, we investigate the effect of initial observation window on the performance of IntentKV. As represented in Table 6, IntentKV can maintain stable performance in a wide range of observation window settings, supporting that it can identify the intention robustly and retain crucial KVs.

D.2 Ablation on Block Mechanism

Table 7: Ablation results of block mechanism of IntentKV and pooling mechanism of SnapKV.

Method	Eff.	4K	8K	16K	32K	64K	128K	Avg.
IntentKV w/o block	8K	95.2	85.9	74.6	59.7	51.0	36.0	67.1
SnapKV w/o pooling	4K	95.2	64.2	61.1	39.3	32.7	20.8	52.2

It is noteworthy that the performance of IntentKV without block mechanism is lower than SnapKV, which we claim to be expected. Since SnapKV utilizes a pooling layer to aggregate information, which can be seen as a "soft" block mechanism. Thus, to present a fair comparison between IntentKV and SnapKV to demonstrate the effectiveness of intention mechanism, we ablate both the block mechanism in IntentKV and pooling mechanism in SnapKV, and the results are summarized in Table 7.

D.3 Ablation on Components of IntentKV

To explicitly illustrate the effectiveness of intention awareness, we ablate the intention mechanism as well as the block mechanism of IntentKV, and the results are presented in Table 8. From this table, we can conclude that both intention and block mechanisms contribute greatly to IntentKV and their orthogonal roles are validated by the higher gain of their combination compared to those of their own.

E More Evaluation on Llama2-7B

E.1 Performance against More Baselines

To evaluate our method against more recent methods, despite the orthogonality, we incorporate additional experiments with Llama-2-7B on LongBench, compared with PyramidKV and CAKE, where the maximum KV cache budget is set to 1024 tokens. Besides, the evaluation can also validate the effectiveness of our method on MHA-based models.

Table 8: Ablation study on intention-awareness and block mechanism of IntentKV.

IntentKV	4K	8K	16K	32K	64K	128K	Avg.
Full	95.2	92.7	90.9	85.7	81.9	75.4	87.0
w/o block	95.2	85.9	74.6	59.7	51.0	36.0	67.1
w/o intent	95.2	88.1	89.9	77.4	72.6	65.8	81.5
w/o both	95.2	64.4	60.4	38.2	32.9	21.4	52.1

From Table 9, we can conclude that all these methods demonstrate strong and comparable performance, with metrics closely approaching those of FullKV, despite variation across tasks. It is worth noting that the main objective of IntentKV, and the baseline we chose in the main experiments, is to **identify important tokens** for the generation, while the key innovation of the methods here is the layer-specific or head-specific **cache budget allocation**. Thus, we believe our method is orthogonal to and can be integrated with these methods to further improve the performance. Besides, we also note that CAKE and PyramidKV took more clock time to perform inference as a results of their layer-specific budget allocation and imbalance computation across layers.

E.2 Efficiency

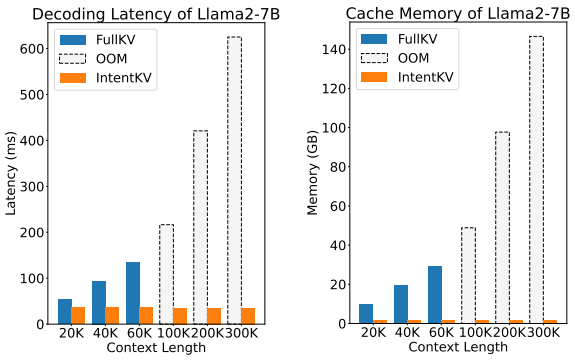


Figure 9: Efficiency evaluation of IntentKV with Llama2-7B on various context lengths.

To provide a more comprehensive evaluation of the efficiency of IntentKV, we conduct a comparative analysis in Llama2-7B that is integrated with multi-head attention. As presented in Fig. 9, IntentKV brings a more substantial reduction in both decoding latency and memory overhead of cache for multi-head attention models. Besides, due to the large reduction of memory occupied by the cache, IntentKV can enable the model to infer much longer sequences without out-of-memory problem.

It is noteworthy that IntentKV achieves consistent efficiency improvement for MHA-based mod-

Table 9: Comparison of more recent and orthogonal KV cache eviction methods on LongBench with Llama2-7B. Best results are highlighted in bold.

Method	NrqQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAmSum	PCount	PR-en	Lcc	R-p	Avg.
FullKV	18.39	20.11	35.67	31.25	25.5	10.14	25.71	20.93	26.27	64	83.38	40.99	5.5	10	60.81	55.29	33.37
PyramidKV	17.32	19.14	**37.19**	30.86	25.3	9.31	22.62	20.6	24.97	**64**	**83.81**	40.65	**6**	**10.5**	**60.86**	54.29	32.96
CAKE	**18.03**	**23.07**	36.47	30.03	**25.71**	8.6	**22.97**	**21.21**	**25.4**	63.5	83.79	**41.15**	2.5	9.5	59.87	**55.96**	32.99
IntentKV	17.45	21.22	35.34	**31.99**	25.59	**9.8**	21.74	20.55	24.42	**64**	83.13	40.44	**6**	**10.5**	60.22	55.66	**33.00**

els at arbitrary context lengths. However, for GQA-based models, the efficiency gain on relatively short contexts can be marginal. Specifically, Llama3.1-8B has 32 attention heads but only 8 key/value heads, so by default only $8 \times$ layers $\times$ length $\times 2$ vectors are stored in the cache. However, all current attention-based KV cache eviction methods (including IntentKV) rely on the full attention logits computed across all 32 heads (since different heads attend to different parts of the context), requiring all $32 \times$ layers $\times$ length $\times 2$ vectors to be stored, which diminishes the efficiency advantage on shorter sequences.

F Results on LongBench and NIAH Test

It is noteworthy that LongBench (Bai et al., 2024) and NIAH (gkamradt, 2024) were proposed some time ago and are subject to potential risks of abuse and data leakage. Current LLMs, though performing well on NIAH, still exhibit large degradation on more complex tasks in Ruler, which expands upon the vanilla NIAH test and provides a more comprehensive evaluation. On the other side, LongBench, with an average length of 10k, is also not challenging for recent LLMs with a 100K+ context window. Besides, InfiniteBench, with its 12 unique tasks, covers a similar domain diversity but provides a greater challenge. Thus, we argue the InfiniteBench and Ruler can already provide a thorough evaluation, and we provide results on LongBench and NIAH for the ease of comparison with previous methods.

As presented in Table 10 and Figure 10, the marginal but consistent improvements of IntentKV in the results support our conjectures, where marginal performance gap confirms that LongBench and NIAH are less challenging for current LLMs, and the consistent performance advantage demonstrates the superiority of IntentKV.

G Discussion on Applications

Motivated by insights from pragmatics, IntentKV is an effective and efficient KV eviction method that accurately evicts KVs irrelevant with the intention of the user, achieving promising performance on trending benchmarks. Besides, except long QA tasks(such book QA or code repository QA) covered in the benchmarks, IntentKV is well suited to retrieval-augmented generation, where a short query exists and a large quantity of documents are first retrieved using it and the final context is composed of the documents as well as the query. In this case, IntentKV can help the LLM focuses on relevant documents, reducing the impact of position bias (Liu et al., 2024; Peysakhovich and Lerer, 2023) and the hallucination resulted from irrelevant documents (Huang et al., 2025).

Table 10: Performance of different KV eviction policies with Llama-3.1-8B-Instruct on LongBench, where the **bold** results indicate the highest value in each row excluding FullKV column. The size of KV cache is set to 512.

Dataset	FullKV	StreamingLLM	SnapKV	IntentKV
2WikiMQA	50.2	44.2	47.9	**49.8**
SAMSum	43.2	41.4	**42.1**	42.0
QMSum	25.4	21.0	23.3	**24.0**
TriviaQA	92.3	88.8	91.2	**92.0**
HotpotQA	54.3	46.3	52.7	**54.7**
RepoBench-P	52.0	49.0	49.7	**50.3**
MultiNews	26.9	23.0	24.0	**24.1**
TREC	73.0	54.5	58.5	**65.0**
Qasper	47.2	24.8	34.4	**41.6**
PassageCount	6.6	**7.5**	7.3	**7.5**
LCC	62.0	59.1	60.9	**61.8**
MuSiQue	32.4	24.6	29.8	**31.6**
MultiFieldQA-en	55.3	33.8	49.7	**53.9**
NarrativeQA	31.1	26.5	**29.7**	29.5
PassageRetrieval-en	100.0	96.5	**99.5**	**99.5**
GovReport	33.9	22.0	23.9	**24.5**
Avg.	49.1	41.4	45.3	**47.0**

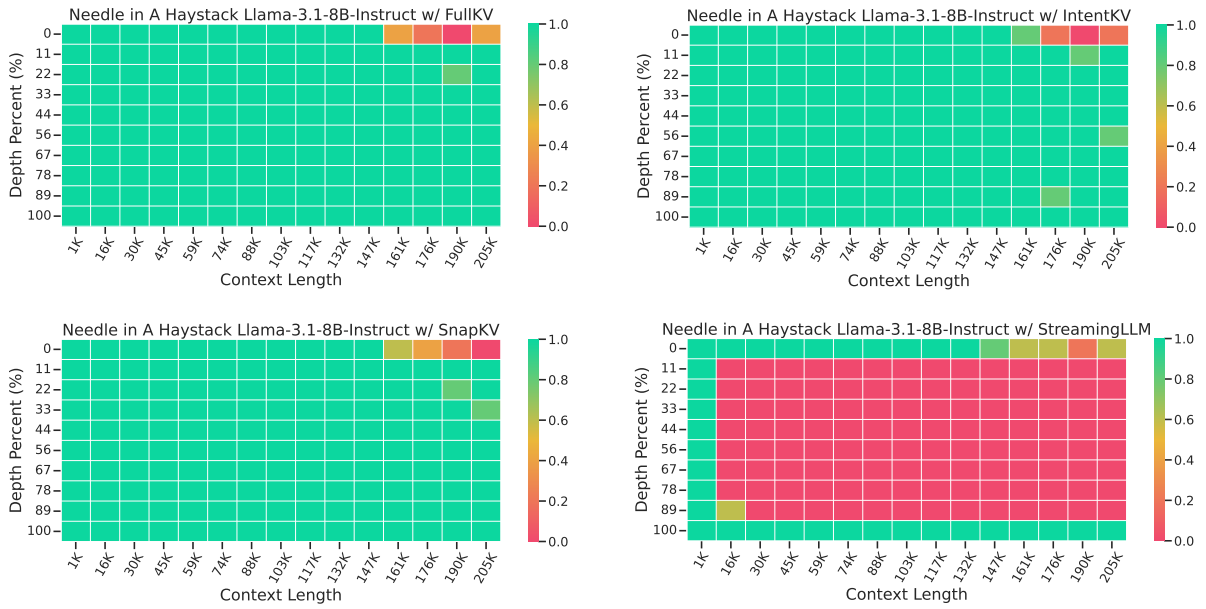


Figure 10: Performance of different KV eviction policies with Llama-3.1-8B-Instruct on NIAH test, where the size of KV cache is set to 2K.