AhaKV: Adaptive Holistic Attention-Driven KV Cache Eviction for Efficient Inference of Large Language Models

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have significantly advanced the field of Artificial Intelligence. However, their deployment is resource-intensive, not only due to the large number of model parameters but also because the (Key-Value) KV cache consumes a lot of memory during inference. While several works propose reducing the KV cache by evicting the unnecessary tokens, these approaches rely on accumulated attention score as eviction score to quantify the importance of the token. We identify the accumulated attention score is biased and it decreases with the position of the tokens in the mathematical expectation. As a result, the retained tokens concentrate on the initial positions, limiting model's access to global contextual information. To address this issue, we propose Adaptive holistic attention KV (AhaKV), it addresses the bias of the accumulated attention score by adaptively tuning the scale of softmax according the expectation of information entropy of attention scores. To make use of the holistic attention information in self-attention mechanism, AhaKV utilize the information of value vectors, which is overlooked in previous works, to refine the adaptive score. We show theoretically that our method is well suited for bias reduction. We deployed AhaKV on different models with a fixed cache budget. Experiments show that AhaKV successfully mitigates bias and retains crucial tokens across global context and achieve state-of-the-art results against other related work on several benchmark tasks.

1 Introduction

Transformer (Vaswani et al., 2017) has demonstrated remarkable success in a wide range of domains, including language modeling (Devlin et al., 2019; Raffel et al., 2020), image recognition (Dosovitskiy, 2021; Xie et al., 2021), and speech recognition (Dong et al., 2018; Karita et al., 2019), owing to its powerful modeling capabilities. In particular, in the field of natural language processing, Transformer-base large language models (LLMs) (Achiam et al., 2023; Zhang et al., 2022; Touvron et al., 2023a,b) have become defacto method. With the rise of multi-turn conversations and long document processing scenarios (Bai et al., 2023; Chen et al., 2024b), the lengths of model's input text (Peng et al., 2024; Liu et al., 2024a) continue to grow, leading to a significant increase in the deployment costs associated with large-scale models. These costs arise not only from the computational resources required for parameter loading and attention computation, but also from the memory consumption caused by the widely used Key-Value (KV) cache strategy during the generation process (Pope et al., 2023). The KV cache which stores precomputed keys and values, which is designed avoid re-computation and improve inference speed. However, this comes at the cost of considerable memory overhead, making it a critical bottleneck in efficient model deployment. For example, in LLaMA-2-7B (Touvron et al., 2023b), when the input batch size is 8 and the context length reaches 32K tokens, the KV cache alone can grow to 128GB, which far exceeds the memory required to store the model parameters themselves.

The substantial KV cache consumption poses significant challenges for deployment on typical consumer GPUs. Recent researches (Xiao et al., 2024; Zhang et al., 2024; Li et al., 2024; Adnan et al., 2024) have shown that retaining only a small subset of crucial tokens' keys and values during generation can achieve performance comparable to those of the full cache LLMs. StreamingLLM (Xiao et al., 2024) identified that initial tokens play a critical role in maintaining model performance. By retaining only the initial tokens and the most recent tokens, StreamingLLM significantly reduces the KV cache size with promising results. To avoid losing vital information within the middle of the input, H_2O (Zhang et al., 2024) leveraged accumu-

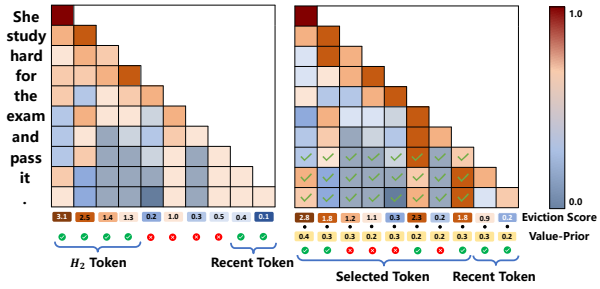


Figure 1: Left: The H_2O-base method, which uses the accumulated attention score as an eviction score, has a higher eviction score for the token on the left side of the sequence. Right: The AhaKV eviction strategy, which focuses the attention scores on key tokens and selects the attention scores of key tokens to calculate the eviction scores, avoiding positional bias.

lated attention scores as the eviction score during generation, enabling the dynamic removal of less important key-value pairs. This flexible and highly successful method has served as the foundation for many research improvements, leading to the development of new approaches. However, in long context tasks, the cumulative attention score is subject to decay bias. SnapKV (Li et al., 2024) uses the recent window's attention score to retain important tokens to improve long context processing. NACL (Chen et al., 2024a) introduces stochastic eviction to reduce the eviction score bias to improve long context performance. However, these methods do not consider the flattening of attention scores in long contexts, and they do not make full use of numerical information.

However, the accumulated attention scores used in existing methods be further optimized. Figure 1 illustrates the attention scores and accumulated attention scores for a segment of a sentence, demonstrating that the H_2O-based approach tends to retain tokens from the earlier part of the sequence. Even when tokens in the middle are semantically crucial, their importance is underestimated within framework. The bias stems from the widely employed causal mask in Transformer-based architectures, which is designed to prevent information leakage from future tokens. Specifically, the causal mask zeros out all values after the current token in the softmax input, ensuring that each token can only attend to itself and preceding tokens. As the sequence length increases, the average attention score for each token decreases after the softmax operation. As depicted in Figure 1, tokens on the right side of the sequence are computed fewer times than those on the left. Furthermore, since tokens on the right inherently receive lower attention scores compared to those on the left during the initial computation, their accumulated attention scores are further diminished in expectation. This results in a systematic bias where tokens on the right side are more likely to be evicted, regardless of their actual importance. Second, the accumulated attention score framework only uses information from the queries and keys, neglecting the information about the values in the self-attention mechanism. In fact, values carry the contextual information prediction. Effectively leveraging the information contained in the values is therefore another key consideration when designing an eviction strategy. By incorporating value-based insights, it may be possible to develop a more balanced and context-aware eviction approach that better preserves semantically important tokens across the entire sequence.

In this paper, we propose Adaptive holistic attention KV (**AhaKV**), which addresses the bias of the accumulated attention score by adaptively tuning the scale of softmax and makes use of the holistic attention information to evict redundant tokens, reducing memory usage while preserving text processing capacity. We analyze the reason of the bias through theoretical derivation, highlighting that a consistent number of cumulative entries is maintained for each token when computing the eviction score. To counteract the flattening of the attention score as the number of tokens increases, we propose setp gain softmax (SG-softmax) to adaptively adjust the attention distribution to emphasize key tokens. Additionally, we compute a prior weight for each token using the modulus length of value to enhance the eviction score. The eviction score in AhaKV avoids positional bias and makes full use of the information of queries, keys, and values (QKV) to maintain the text processing capacity after eviction of the token, and achieves state-of-the-art in comparison to other eviction methods. Overall, Our main contributions are as follows:

(1) Analyses of self-attention mechanisms explain why cumulative attention scores retain early tokens and provide a theoretical basis for addressing bias. Meanwhile, we propose Recent Accumulation and SG-softmax to avoid bias.

(2) Our work pioneers the effective use of value matrix information through a simple transformation that enhances the eviction score using

the value prior. This provides a possible idea for future improvements.

2 Related Work

2.1 Efficient LLMs Inference

LLMs have a large inference computational cost. This problem has inspired a lot of research in the inference optimization for LLMs, such as the methods (Frantar and Alistarh, 2023; Ma et al., 2023; Sun et al., 2024) adopting pruning to reduce the redundant parameters of LLMs and thus improve the inference speed. In addition to this, many research (Frantar et al., 2023; Dettmers et al., 2022; Liu et al., 2023) have used quantization for LLMs to reduce the full-precision LLM parameters to those of a few bits, thus speeding up the inference of the model. In contrast to the above approaches of modifying model parameters, some other works (Zaheer et al., 2020; Wang et al., 2021; Zhang et al., 2024; Hooper et al., 2024; Liu et al., 2024c; Dong et al., 2024) start from the bottleneck of the attention module of secondary computational complexity in inference and speed up the inference process by sparse attention. The approaches of these works are orthogonal and can be integrated together. The work studied in this paper, on the other hand, is closely related to sparse attention and focuses on reducing the memory bottleneck, KV cache, that arises in the inference process.

2.2 Sparse Attention

Sparse attention mainly targets the prefill phase of the transformer and omits certain attention operations to improve computational efficiency. Big bird (Zaheer et al., 2020) combines random, local and global attention to sparse attention while maintaining accuracy of generation. Longformer (Beltagy et al., 2020) introduces sliding windows and task-driven global attention to reduce attention computation. Spatten (Wang et al., 2021) evaluates the importance of each token by calculating the cumulative attention to dynamically prune the token with minimal attention. However, these sparse-attention approaches do not focus on the growing memory problem despite reducing self-attention computation. The KV cache generated by LLMs inference has become a memory bottleneck for inference.

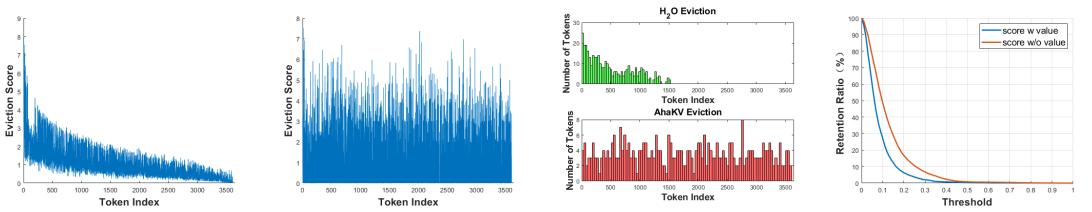
2.3 KV Cache Eviction

Research on KV cache optimization has focused on compressing the out of memory that occurs as the increasingly length of the prompt. In this paper, we focus on KV cache eviction (Xiao et al., 2024; Zhang et al., 2024; Chen et al., 2024a; Adnan et al., 2024) which focuses on attention sparsity, and explore how to reduce redundantly labelled KVs to reduce memory usage and improve inference efficiency. Which can be subdivided into static eviction and dynamic eviction according to the eviction method. The static eviction is represented by StreamingLLM (Xiao et al., 2024), which observes that the beginning token is very important for the whole generation task, and therefore retains the KVs of the 4 initial tokens and the most recent token, and evicts the KVs of the remaining tokens. The study of dynamic eviction focuses on identifying the key tokens and constantly updating the KV cache. The H_2O (Zhang et al., 2024) method utilises the sum of cumulative attentions as the eviction score and retains the token with the highest score and the nearest token. SnapKV (Li et al., 2024) proposed to compress the long context prompt before calculating the expulsion score to improve the comprehension of long text. NACL (Chen et al., 2024a) introduces random evictions thereby avoid bias in eviction scores. Distinguishing from the above methods, AhaKV avoids the bias of the Eviction Score and makes full use of value to refine the eviction score.

3 Motivation

For Decoder-only Large Language Models, the inference process consists of two phases, the prefill phase and the generation phase. In the prefill phase, input tokens are encoded as Key, Value, and Query vectors to compute the self-attention. The Key and Value vectors are typically cached for reusage to accelerate inference. In the generation phase, the model computes attention weights by multiplying the current token's Query with all cached Key vectors, then applies these weights to the corresponding Value vectors. With KV caching, only the Key and Value of the new token are computed at each step, while previous vectors are retrieved from the cache. While this approach reduces computational redundancy, it incurs significant memory overhead.

Attention computations exhibit inherent sparsity in practice. Selective eviction of non-critical Key-Value pairs from the cache can reduce memory usage while preserving generation accuracy (Zhang et al., 2024; Liu et al., 2024b; Adnan et al., 2024). Current eviction strategies predominantly rely on



(a) H_2O (b) AhaKV (c) Retain token index. (d) Sparsity Comparision

Figure 2: (a,b) shows the eviction scores in different strategy. (c) shows the retention token index. (d) shows the Comparision between Rretention Ratio in different threshold with and without value-prior

accumulated attention scores (Zhang et al., 2024). However, we identify two critical limitations in these approaches.

3.1 Positional Bias in Accumulated Attention Scores

Figure 2 analyzes the eviction scores generated by H_2O (Zhang et al., 2024) for an attention head of a layer in the LLaMA model, along with the distribution of retained token indices across a 3,600-token sequence. The results reveal an intrinsic positional bias: accumulated attention score has a intrinsic bias, that they gradually decrease with token position. Tokens beyond position 1,500 are disproportionately evicted, leading to the catastrophic loss of nearly half of the context information. It highlights the eviction strategies need to ensure more uniform information retention across all positions.

3.2 Neglecting Values' Effect

While Value vectors fundamentally shape self-attention outputs, existing eviction strategies ignore their contribution. We propose augmenting accumulated attention scores with Value-derived prior weights to refine eviction scores generated by H_2O. Figure 2(d) compares normalized retention rates for thresholds applied to baseline scores versus value-enhanced scores. Incorporating value priors obtains lower retained token proportion at equivalent thresholds. This demonstrates that value magnitudes provide complementary signals for identifying semantically important tokens.

4 Methodology

In this section, we analyze the limitations of accumulated attention scores and the computation of softmax mathematically. We identify the reason of the bias and mitigate it through theoretical deduction. We further refine the eviction score using the value-based prior to make use of the holistic attention. A novel method Adaptive holistic attention KV (AhaKV) that evicts tokens based on unbiased eviction scores while preserving global context information is proposed.

4.1 Analysis of the Accumulated Attention Score

Denote the query matrix as $Q \in \mathbb{R}^{n\times d}$, the key matrix as $K \in \mathbb{R}^{n\times d}$, and the hidden dimension as d. The attention score between the i^{th} query and the j^{th} key $a_{i,j}$ is computed as following:

$$a_{i,j} = \text{softmax}(Q_i^T K_j/\sqrt{d}) = \frac{e^{Q_i^T K_j/\sqrt{d}}}{\sum_{j=1}^{i} e^{Q_i^T K_j/\sqrt{d}}} \tag{1}$$

The eviction score S_j in H_2O for the j^{th} token is defined as:

$$S_j = \sum_{i=1}^{n} a_{i,j} \tag{2}$$

Due to the casual mask, $a_{i,j} = 0$ when $i < j$. Thus, Eq. (2) can be rewritten as:

$$S_j = \sum_{i=j}^{n} a_{i,j} \tag{3}$$

The number of cumulated term decreases as j increases, leading to a monotonic decrease trend in the eviction score, which will be proven in the following. Following the previous works (Vaswani et al., 2017; Chi et al., 2023; Kazemnejad et al., 2024), we assume that the components of Q_i and K_j are independent random variables with the mean as 0 and the variance as 1. Under this assumption, $a_{i,j}$ has the same expectation and variance for the same row of the attention matrix. We prove the

monotonicity of the eviction score as follows:

$$
\begin{aligned}
&E(S_{j+1} - S_j) = E(S_{j+1}) - E(S_j) \\
=&E(\sum_{i=j+1}^{n} a_{i,j+1}) - E(\sum_{i=j}^{n} a_{i,j}) \\
=&\sum_{i=j+1}^{n} E(a_{i,j+1}) - \sum_{i=j}^{n} E(a_{i,j}) \\
=&\sum_{i=j+1}^{n} [E(a_{i,j+1}) - E(a_{i,j})] - E(a_{j,j}) \qquad (4)
\end{aligned}
$$

Since each element in row i has the same mathematical expectation, the first term of Eq. (4) is 0. However, the expectation of $a_{j,j}$ is a positive value. Thus, $E(S_{j+1} - S_j) < 0$, which means that the eviction score decreases monotonically in expectation. This is validated empirically in Figure 2, where the accumulated attention score has an clear positional bias.

To address this bias, we propose summing the attention scores over the nearest r rows. This ensures that the eviction score is not influenced by the number of accumulated terms. The modified eviction score is computed as:

$$S_j = \sum_{i=r}^{n} a_{i,j} \qquad (5)$$

4.2 Rethink the Softmax Computation

We further analyze the decrease in attention scores as the sequence length grows. As more tokens are included in the softmax computation, the attention scores become diluted, particularly for tokens in later positions. This phenomenon arises because the softmax normalization distributes a fixed total probability mass (summing to 1) across a growing number of tokens, leading to smaller average attention scores for each token. To better understand the distribution of attention scores, we introduce information entropy from information theory as a complementary measure. Let H_i be the information entropy of the i^{th} row of the attention score matrix, which is given by

$$H_i = -\sum_{j=0}^{i} a_{i,j} \log a_{i,j} \qquad (6)$$

Information entropy quantifies the concentration or dispersion of the attention score distribution. A higher entropy value indicates a smoother, more uniform distribution. We will analyze the variation of H_i as the number of rows i increases. Replacing $Q_i^T K_j$ with $w_{i,j}$ for simplification and substituting Eq. (1) into Eq. (6), we have

$$H_i = \log \sum_{j=0}^{i} e^{w_{i,j}} - \frac{\sum_{j=0}^{i-1} e^{w_{i,j}} \cdot w_{i,j}}{\sum_{j=0}^{i-1} e^{w_{i,j}}} \qquad (7)$$

We further assume that $w_{i,j}$ follows the Gaussian distribution. Then, we can deduct that the expectation of H_i is

$$E[H_i] = \log i - \frac{d}{2}. \qquad (8)$$

Details of the deduction can be found in the appendix A. Eq. (8) shows that expectation of H_i increases with i. This indicates that as the number of tokens grows, the attention score distribution becomes flatter, and the maximum attention score decreases. Consequently, tokens in later positions receive lower attention scores overall. To tackle this, we introduce a scaling parameter λ to adaptively adjust the smoothness of the distribution according to the number of tokens. Mathematically, we transform the softmax function from $\text{softmax}(x_i) = \frac{x_i}{\sum_{i=0}^{n} e^{x_i}}$ to step gain sofmax

$$\text{SG-softmax}(x_i, \lambda_i) = \frac{\lambda x_i}{\sum_{i=0}^{n} e^{\lambda x_i}}. \qquad (9)$$

Then, the expectation of H_i changes into

$$E[H_i] = \log i - \frac{\lambda^2 d}{2}. \qquad (10)$$

Assume that the token number of the budget is k. The ideal case for a good eviction score is that the there are only k attention score larger than zero. We aim to make expected information entropy equals to the maximum information entropy regardless of the number of total tokens. Therefore, we have

$$\log i - \frac{\lambda^2 d}{2} = -\sum_{j=0}^{k} \frac{1}{k} \log \frac{1}{k} = -\log \frac{1}{k}. \qquad (11)$$

We can figure out that, when the number of total tokens is i,

$$\lambda = \sqrt{\frac{2\log(i/k)}{d}}. \qquad (12)$$

Eq. (12) gives an estimation for the scaling parameter λ in practice.

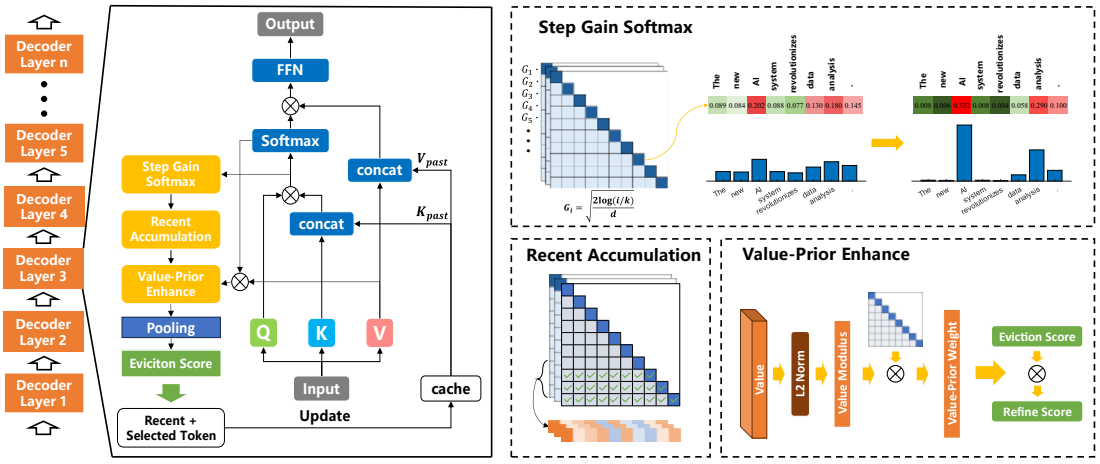


Figure 3: AhaKV consists of three main components: SG-softmax, Recent Accumulation, and Value-Prior Refine. First, SG-softmax replaces the standard softmax to prevent attention score flattening. Then, cumulative eviction scores are computed using attention scores from recent rows to ensure each token has the same number of cumulative terms. Next, value-prior is used to refine eviction score selection, and pooling is applied to smooth scores and prevent excessive text sparsification. Finally, tokens are retained based on their eviction scores.

4.3 Enhance the Eviction Score via Value-prior

Through the steps outlined above, we have derived unbiased eviction scores. However, since both attention scores and value vectors play a critical role in the computation of self-attention, it is essential to account for their importance. In order to utilize the holistic attention information, we refine the accumulated attention scores by incorporating the magnitude of the value vectors.

Let $V \in \mathbb{R}^{n \times d}$ denote the value matrix. The square of the L_2 norm of the i^{th} token is computed $\nu_i = \|V_i\|^2$. However, in self-attention computation, the features of the current token are obtained by weighting the value vectors using the corresponding attention scores. To reflect this, we use attention scores to computed the weighted sum of ν_i that reflecting the importance of the i^{th} token:

$$\gamma_i = \sum_{j=1}^{i} a_{i,j} \cdot \nu_i. \tag{13}$$

We then normalize it with the maximum value among all γ_i.

$$\bar{\gamma}_i = \frac{\gamma_i}{\max(\gamma)}. \tag{14}$$

Unlike the accumulated attention score which measures token importance based on inter-token correlations, $\bar{\gamma}_i$ captures the importance derived from the model's parameterized weighting of tokens. We use it as a prior weight to refine adaptive accumulated attention score:

$$\hat{S}_i = \bar{\gamma}_i \cdot S_i \tag{15}$$

The details of the AhaKV algorithm are shown in Algorithm 1.

5 Experiments

5.1 Setup and Baseline

Our experiment are conducted based on 3 widely used basic models, LLaMA (Meta, 2024), Qwen (Qwen, 2024) and Gemma(Google, 2024). In order to demonstrate the capability of AhaKV under long text, we chose the LongBench framework (Bai et al., 2023) for testing. There are 21 datasets in six areas in the LongBench framework, including math, code completion, few-shot learning, multi-document QA, single-document QA, summarization and synthetic tasks. By testing 21 datasets within the LongBench framework, we can effectively demonstrate the efficacy of the AhaKV method. In addition to testing within the LongBench framework, we selected the ARC-E (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), WiC (Pilehvar and Camacho-Collados, 2019), and WinoGrande (Sakaguchi et al., 2020) datasets, demonstrating that AhaKV is effective across a diverse range of datasets. Results were averaged over multiple seed results. All experiments are conducted with NVIDIA A800 80GB GPU.

Table 1: Results in Longbench Datasets

	Method	Code Complete	Few Shot	Single-doc QA	Multi-doc QA	Summarization	Passage Retrieval	Average
LLaMA3-8B-Inst	Full Cache	54.47	57.75	41.55	32.33	20.14	54.83	41.94
	Sink(Xiao et al., 2024)	56.59	53.03	30.31	26.88	17.75	26.50	33.55
	H_2O(Zhang et al., 2024)	**58.88**	54.57	38.61	30.76	**19.75**	41.70	38.93
	SnapKV(Li et al., 2024)	56.52	56.36	40.56	31.06	18.22	54.33	40.99
	NACL(Chen et al., 2024a)	56.62	54.78	40.32	31.39	18.49	54.33	40.77
	TOVA(Oren et al., 2024)	53.45	**57.29**	38.6	30.74	17.63	53.30	40.18
	AhaKV	58.20	57.08	**41.18**	**31.82**	18.74	**54.83**	**41.74**
Qwen2-7B-Inst	Full Cache	59.35	61.60	45.20	34.35	22.65	39.33	42.47
	Sink(Xiao et al., 2024)	56.09	55.41	29.56	26.45	19.43	15.33	32.46
	H_2O(Zhang et al., 2024)	52.72	56.14	37.82	31.77	21.15	22.67	36.24
	SnapKV(Li et al., 2024)	**59.08**	60.37	43.99	33.33	20.71	**38.50**	41.30
	NACL(Chen et al., 2024a)	58.67	53.67	41.80	32.02	20.83	38.00	39.27
	TOVA(Oren et al., 2024)	56.91	56.65	37.97	30.69	19.00	35.61	37.99
	AhaKV	59.01	**61.01**	**44.24**	**33.99**	**22.12**	38.33	**41.83**
LLAMA2-7B-Chat	Full Cache	55.72	51.99	22.28	18.56	18.55	5.34	27.28
	Sink(Xiao et al., 2024)	53.35	49.26	14.11	15.01	15.66	1.94	23.27
	H_2O(Zhang et al., 2024)	42.66	49.32	20.46	16.29	**18.27**	4.01	24.51
	SnapKV(Li et al., 2024)	55.32	50.95	21.50	17.65	16.82	5.57	26.39
	NACL(Chen et al., 2024a)	54.50	49.84	19.92	17.82	16.97	**6.54**	26.04
	TOVA(Oren et al., 2024)	53.68	51.16	16.10	16.41	15.68	4.34	24.66
	AhaKV	**55.98**	**51.49**	**22.42**	**17.98**	17.11	5.57	**26.89**
Gemma-7B-Inst	Full Cache	46.91	52.75	33.82	23.81	20.47	27.02	33.25
	Sink(Xiao et al., 2024)	48.84	49.78	20.34	18.63	18.05	15.33	27.18
	H_2O(Zhang et al., 2024)	50.01	49.99	29.88	21.29	**20.13**	22.55	31.09
	SnapKV(Li et al., 2024)	47.66	52.45	32.89	22.35	18.84	26.29	32.39
	NACL(Chen et al., 2024a)	47.09	50.35	31.81	21.93	19.12	26.69	31.77
	TOVA(Oren et al., 2024)	45.69	49.20	30.61	21.20	18.78	25.38	30.80
	AhaKV	**50.44**	**53.37**	**33.05**	**22.89**	19.38	**26.91**	**33.16**

Algorithm 1 AhaKV Algorithm

Total Cache Budget B($B = B_r + B_s$), Recent Cache Budget B_r, Selected Cache Budget B_s, Temperature β
function PREFILL($Prompt$)
  for Layer-i in LLMs **do**
    for Head-m in Layers **do**
      $Q_{m,i}, K_{m,i}, V_{m,i} \in \mathbb{R}^{n \times d}$
      $A_{m,i} \leftarrow \frac{Q_{m,i} \cdot K_{m,i}^T}{\sqrt{d}}$
      $S^r_{m,i} \leftarrow$ Recent B_r Token
      $F_s \leftarrow \sum_{Token \in S^r_{m,i}} SGsoftmax(A, \beta)$
      $\hat{F}_s \leftarrow \bar{\gamma} \cdot F_s$ ▷ In Section 4.3
      $S^s_{m,i} \leftarrow TopK(F_s, B_s)$
      $S^{Prefill}_{m,i} \leftarrow S^s_{m,i} \bigcup S^r_{m,i}$
    end for
  end for
end function
function GENERATION($S^{Prefill}_{m,i}, MaxLength$)
  $S^0 \leftarrow S^{prefill}$
  $Z_0 \leftarrow$ First Token
  $F_0 \leftarrow \hat{F}_s$
  for $t < MaxLength$ **do**
    for Layer-i in LLMs **do**
      for Head-m in Layers **do**
        $K^{t-1}_{m,i}, V^{t-1}_{m,i} \leftarrow K_{S^{t-1}_{m,i}}, V_{S^{t-1}_{m,i}}$
        $K^t_{m,i} \leftarrow (K^{t-1}_{m,i}, W^K_{m,i} Z_0)$
        $V^t_{m,i} \leftarrow (V^{t-1}_{m,i}, W^V_{m,i} Z_0)$
        $A_{m,i} \leftarrow \frac{W^Q_{m,i} Z_{0_{m,i}} \cdot K^T_{m,i}}{\sqrt{d}}$
        $F_t \leftarrow SGsoftmax(A, \beta) + F_{t-1}$
        $S^r_{m,i} \leftarrow$ Recent B_r Token
        $S^s_{m,i} \leftarrow TopK(F_s, B_s)$
        $S^{Prefill}_{m,i} \leftarrow S^s_{m,i} \bigcup S^r_{m,i}$
      end for
    end for
  end for
end function

We select H2O(Zhang et al., 2024) as the baseline and primary improvement method. H2O use accumulated attention score and discards the lower value. In addition, we compare AhaKV with several popular eviction strategies, including Sink (Xiao et al., 2024), SnapKV (Li et al., 2024), NACL (Chen et al., 2024a) and TOVA (Oren et al., 2024).

5.2 Comparison Results

5.2.1 Longbench Results

We conducted extensive testing on the LongBench dataset, which consists of six categories of tasks as in Table 1. LLAMA2-7B was assigned a total budget of 720 due to its limited sequence length, while other models received 1000, with a fixed recent budget of 32 applied uniformly across all models. We use Eq. 12 to initialize the parameter λ. As shown in Table 1, AhaKV achieved state-of-the-art performance, attaining the highest average accuracy across all tasks.

The AhaKV method demonstrates significant performance advantages across multiple benchmarks. In code-completion tasks, it achieved superior accuracy over the full cache baseline by 0.2% and 3.53% on LLAMA2-7B-Chat and Gemma-7B-Inst, respectively. For few-shot learning scenarios, AhaKV outperformed all competitors on Gemma-7B-Inst with a 0.92% lead over the second-best method, though trailing slightly on LLaMA3-8B-Inst. It consistently dominated both single-document and multi-document QA tasks across all tested models, notably surpassing the full cache baseline by 0.14% on LLAMA2-7B-Chat in single-

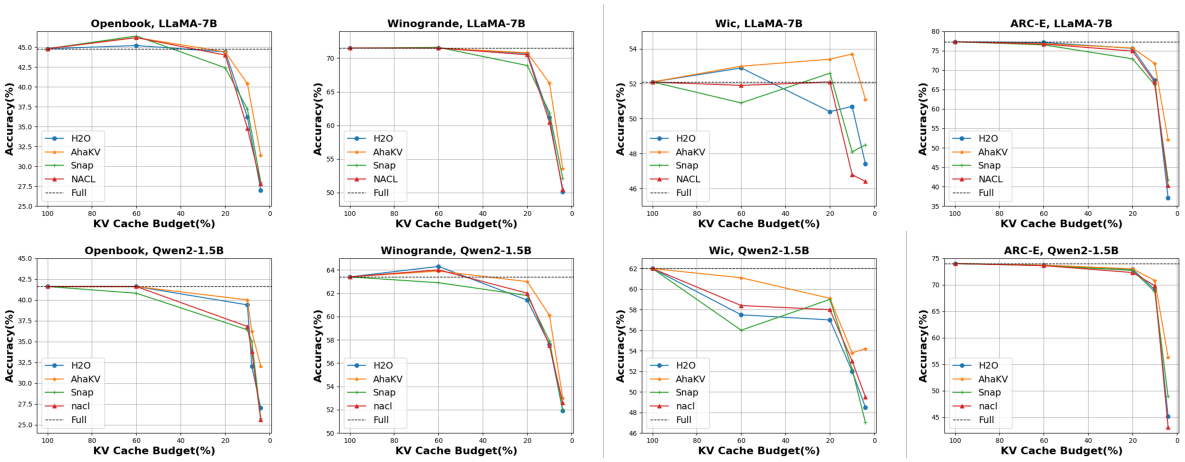


Figure 4: Short-text results in different KV budget.

document QA. The method enhances summarization through context-aware extraction of highly relevant tokens and shows strong retrieval capabilities, outperforming rivals by 0.5% and 0.22% on LLaMA3-8B-Inst and Gemma-7B-Inst in passage retrieval. These key metrics demonstrate AhaKV's consistent performance advantages across diverse tasks and model architectures.

5.2.2 Short-text Results

To validate the effectiveness of AhaKV in processing short texts, we selected the OpenBook, ARC-E, WiC, and WinoGrande datasets for testing and comparison against H2O(Zhang et al., 2024) and other methods in Figure 4.

AhaKV has been evaluated using LLAMA2-7B and Qwen2-1.5B, with results obtained for a KV Budget of 60%-4%. The experiments demonstrate that AhaKV can achieve exceptional performance with short text datasets. Notably, as the compression rate increases (resulting in a decreased Budget), AhaKV shows a tendency to deliver higher accuracy compared to H2O and others.

Compared to other methods, SampkeKV consistently outperforms other expulsion schemes in terms of accuracy in the most extreme compressed case (only 4% budget). It shows that AhaKV is able to give a high enough weight to crucial tokens in the text to make them less likely to be evicted.

Table 2: Ablation study in subdatasets in LongBench of llama2-7B.

Datasets	2Wiki	TriQA	Multi-EN	lcc	Samsum	Avg
w/o RA	28.7	84.17	31.01	57.38	39.49	48.15
w/o SGS	29.78	81.23	29.67	50.32	40.37	46.25
w/o VPE	29.54	82.94	33.96	58.16	41.06	49.13
AhaKV	29.96	83.61	34.6	58.4	40.55	49.42

5.3 Ablation Study

In AhaKV, we introduce three ways to improve eviction strategies: **Recent Accumulation**, **Step Gain Softmax** and **Value-Prior Enhance**. To evaluate the orthogonality of the three components and assess their individual impact, we conducted ablation experiments on LLAMA2-7B-Chat, examining each part separately. To simplify the presentation, we use the abbreviations RA for Recent Accumulation, SGS for Step Gain Softmax, and VPE for Value-Prior Enhance in the table.

Table 2 presents ablation results for LLAMA2-7B-Chat on LongBench subsets (one dataset per category). The principal findings are outlined as follows. (i) Removing RA caused a 1.27% average accuracy drop, confirming the necessity of stable cumulative terms to reduce bias. (ii) Eliminating SGS reduced accuracy by 3.17%, demonstrating its critical role in eviction strategy enhancement. (iii) Disabling VPE decreased accuracy by 0.29%, proving its effectiveness in refining eviction scores to prioritize essential tokens.

6 Conclusion

In this paper, we address the problem of KV cache compression by proposing AhaKV, a KV cache eviction algorithm that can identify important tokens and reason with a fixed KV cache budget, solving the problem of KV cache growth at inference. Our approach selects key tokens based on the attention score, and proposes Step Gain Softmax and Sample Attention to eliminate the attention bias problem caused by causal mask. Experimental evaluations show that AhaKV achieves good performance on both short text and long text datasets.

Limitations

The main limitations of the approach presented in the paper are the following: firstly, due to resource constraints, we did not conduct our experiments on longer texts. However, we have derived our method and evaluated our model on datasets of different lengths in the paper, and we believe that AhaKV can be adapted to reasoning with longer texts. Second, for the choice of a constant number of accumulators in prefill, we choose the recent rows in the attention score for accumulation; in practice, we do not think this choice is unique. For example, we could also choose the Topk strategy to select the same number of cumulants.

Ethics Statement

In this study, we utilize open-source data and technologies, which significantly mitigate privacy concerns. With the polish of the AI assistant, we more clearly define the problem and illustrate the approach. Our novel approach focuses on enhancing the understanding of model contexts and improving inference efficiency, with the goal of creating accessible and highly effective models capable of handling extended contexts. This strategy is expected to advance the openness of NLP technologies and facilitate their practical deployment in diverse applications. Crucially, our methodology is independent of the training process, ensuring that it does not perpetuate or introduce biases into the models. By emphasizing state-of-the-art, resource-efficient techniques, our research contributes to the development of a more open and automated AI, pushing the boundaries of artificial intelligence while ensuring that the benefits of these advancements are widely accessible and applicable across various fields. This represents a significant step toward a more inclusive and automated AI-driven future.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant Nair, Ilya Soloveychik, and Purushotham Kamath. 2024. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of the Seventh Annual Conference on Machine Learning and Systems*, 6:114–127.

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. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 3119–3137.

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

Yilong Chen, Guoxia Wang, Junyuan Shang, Shiyao Cui, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, Yu Sun, Dianhai Yu, and Hua Wu. 2024a. NACL: A general and effective KV cache eviction framework for LLM at inference time. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7913–7926.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024b. Longlora: Efficient fine-tuning of long-context large language models. *International Conference on Learning Representations*.

Ta-Chung Chi, Ting-Han Fan, Li-Wei Chen, Alexander Rudnicky, and Peter Ramadge. 2023. Latent positional information is in the self-attention variance of transformer language models without positional embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1183–1193.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3.int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, volume 1, pages 4171–4186.

Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. 2024. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. *International Conference on Machine Learning*.

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: a no-recurrence sequence-to-sequence

model for speech recognition. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5884–5888.

Alexey Dosovitskiy. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*.

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.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate post-training compression for generative pretrained transformers. *International Conference on Learning Representations*.

Google. 2024. Gemma. https://github.com/google-deepmind/gemma.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*.

Shigeki Karita, Nelson Enrique Yalta Soplin, Shinji Watanabe, Marc Delcroix, Atsunori Ogawa, and Tomohiro Nakatani. 2019. Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration. In *Annual Conference of the International Speech Communication Association*, volume 2019, pages 1408–1412.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2024. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36.

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. *Advances in Neural Information Processing Systems*.

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. 2024a. World model on million-length video and language with ringattention. *arXiv preprint arXiv:2402.08268*.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. Llm-qat: Data-free quantization aware training for large language models. *Findings of the Association for Computational Linguistics*, pages 467–484.

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

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024c. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *International Conference on Machine Learning*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

Meta. 2024. Llama. https://www.llama.com/.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for openbook question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391.

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

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. Yarn: Efficient context window extension of large language models. *International Conference on Learning Representations*.

Mohammad Taher Pilehvar and José Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1267–1273. Association for Computational Linguistics.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624.

Qwen. 2024. Qwen. https://huggingface.co/Qwen.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 8732–8740.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. *International Conference on Learning Representations*.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, pages 5998–6008.

Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. *International Conference on Learning Representations*.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. 2021. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

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

A Derivation of the mathematical expectation of H_i

The mathematical expectation of H_i is shown as Eq.16

$$logi + logE[e^{w_{i,j}}] - \frac{E[e^{w_{i,j}} \cdot w_{i,j}]}{E[e^{w_{i,j}}]} \tag{16}$$

Let $x = w_{i,j}$ and assume that x is a Gaussian distribution with mean μ and variance σ^2. Symbolically, $x \sim N(\mu, \sigma^2)$. Its probability density function is shown as Eq. 17.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{17}$$

The mathematical expectation of e^x in Gaussian distribution is shown as the Eq. 18.

$$E(e^x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^x \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{18}$$

Combine the Equation 18 to the Equation 19.

$$E(e^x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{x-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{19}$$

To simplify the Equation 19, we adjust the $x - \frac{(x-\mu)^2}{2\sigma^2}$ as follow.

$$\mu + \frac{\sigma^2}{2} - \frac{[(x-\sigma^2)-\mu]^2}{2\sigma^2} \tag{20}$$

Combine the Equations 19 and 20 to Equation 21.

$$e^{\mu+\frac{\sigma^2}{2}} \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[(x-\sigma^2)-\mu]^2}{2\sigma^2}} dx \tag{21}$$

Denote t as $x - \sigma^2$, the Equation 21 could be transformed as the Equation 22.

$$e^{\mu+\frac{\sigma^2}{2}} \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt \tag{22}$$

The $\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dx$ is the cumulative distribution function of random variable which obey the Gaussian distribution with mean μ and variance σ. The result of the integral is 1. Thus, the mathematical expectation of e^x is represented as follow.

$$E(e^x) = e^{\mu+\frac{\sigma^2}{2}} \tag{23}$$

The computation of $E(xe^x)$ is shown as the Equation 24.

$$E(xe^x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} xe^x \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \quad (24)$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} x \cdot e^{x-\frac{(x-\mu)^2}{2\sigma^2}} dx \quad (25)$$

To simplify the Equation 25, we adjust the $x - \frac{(x-\mu)^2}{2\sigma^2}$ as shown in the Equation 20. Thus, the Euqation 25 is transformed into the Equation 26.

$$e^{\mu+\frac{\sigma^2}{2}} \int_{-\infty}^{\infty} \frac{x}{\sigma\sqrt{2\pi}} e^{-\frac{[(x-\sigma^2)-\mu]^2}{2\sigma^2}} dx \quad (26)$$

Denote u as $x - \sigma^2 - \mu$, the Equation 26 is transformed into the Equation 27.

$$\frac{e^{\mu+\frac{\sigma^2}{2}}}{\sigma\sqrt{2\pi}} [\int_{-\infty}^{\infty} (\sigma^2+\mu) e^{-\frac{u^2}{2\sigma^2}} du + \int_{-\infty}^{\infty} ue^{-\frac{u^2}{2\sigma^2}} du] \quad (27)$$

The first term of the Equation 27 is similar to the integral of the Gaussian distribution and its result is $\sigma\sqrt{2\pi}$. The second term of the Equation 27 is an odd function and its integral is 0. In summary, the result of the Equation 27 is shown as follow.

$$E(xe^x) = e^{\frac{\sigma^2}{2}+\mu}(\mu+\sigma^2) \quad (28)$$

Combining the above derivations, we can obtain the result of Eq. A as shown in Eq. 29.

$$\log i - \frac{d}{2} \quad (29)$$