Memory Optimization in Language Model Inference through Key-Value Cache Pruning

Zhen Sun 31520231154312[1], Hongwei Niu 36920231153225[2], Pengfei Yue 36920231153259[2], Haoqi Zhang 36920231153260[3], Zhaohong Huang 36920231153197[1] Wenhao Li 36920231153209[1]

[1]Department of Artificial Intelligence, School of Informatics, XMU
[2]AI Class 1, Institute of Artificial Intelligence, XMU
[3]AI Class 2, Institute of Artificial Intelligence, XMU

Abstract

This paper discusses the issue of memory consumption in key-value (KV) caches during the language model inference process. To address this problem, we propose a pruning method that utilizes attention scores as a metric to evaluate the importance of each key. We introduce transformations to the attention scores to correct statistical bias and emphasize the importance of incorporating a forgetting factor into the process. Statistical bias correction involves dividing the columns of the attention score matrix by a factor related to the number of accumulated scores and multiplying the rows by a factor related to the position of the query. This modification ensures that the expected score for each key is 1, thereby mitigating the bias caused by varying numbers of accumulated scores. The forgetting factor mechanism focuses on recent query scores, capturing local dependencies, and reducing computational overhead.Additionally, we introduce an innovative quantization method that dynamically adjusts the quantization precision based on the relevance of keys (K) and values (V) vectors to the query (Q). The most relevant keys are quantized with high precision, while less relevant or irrelevant keys are compressed using lower resolutions. This dynamic quantization process has the potential to improve the efficiency of KV caches in large models. Experimental evaluations will be conducted to validate the effectiveness of the proposed pruning and quantization methods. These improvements aim to optimize the memory consumption of KV caches in language model inference, thereby enhancing the overall performance of language models.

Introduction

Introduction and Challenges of KV Cache

KV cache servers are a specialized type of high-speed caching servers designed to store data in memory and quickly respond to frequently accessed data requests. The term "KV" refers to the key-value model that these servers employ, offering efficient data storage and fast read/write access capabilities. They find extensive usage in distributed systems, web applications, and large-scale enterprise applications.

In the field of natural language processing, training and inference with language models often require substantial computational resources and time. To enhance the efficiency of these models, Large Language Models (LLMs) incorporate the use of KV cache mechanisms. KV cache, acting as a key-value database, is utilized to store pre-calculated contexts, responses, their representations, and computational outcomes. In LLM models, when encountering a new context or generating a response, the model stores the computed results in the KV cache. Subsequently, when future contexts or responses match the cached keys, the model can retrieve the pre-calculated results directly from the KV cache, thereby avoiding the need for recomputation.

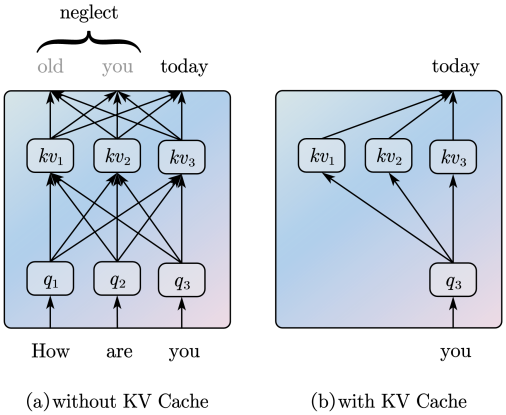

Figure 1: The mechanism of KV cache.

KV cache reduces redundant computations by storing and reusing results, thereby greatly enhancing the efficiency of training and inference processes, as 1 shows.

This approach helps conserve computational resources and reduces time requirements. In the case of dialogue systems and other tasks involving frequent interactions and multi-turn conversations, KV cache plays a crucial role. By rapidly providing access to previously computed results, it significantly accelerates the system's response time, improving the overall user experience.

However, the drawbacks of KV cache become apparent in certain aspects. Firstly, it necessitates a certain amount of memory space to store previously computed results. If the volume of cached data is substantial, it can require a significant amount of memory resources. Additionally, the con-

tents of the KV cache are based on past input data and the corresponding computed results. If the input data changes but the KV cache is not promptly updated, inconsistencies may arise between the cached data and the actual situation. This article focuses on addressing the challenge of excessive memory usage associated with KV cache.

Current Solutions

Pruning and quantization are common techniques for reducing KV cache.

Pruning is a method that reduces the model's size and memory footprint by removing redundant or unnecessary parameters or computation results. For KV cache, pruning can reduce memory usage by removing infrequently used or relatively unimportant key-value pairs. This retains only the most useful results, reducing memory consumption.

Quantization is the process of converting parameters or computation results from floating-point representations to lower-precision fixed-point or integer representations. For KV cache, quantization can convert computation results represented as floating-point numbers into smaller integer or fixed-point representations. This conversion can significantly reduce the number of bits needed for storage, thus lowering memory usage.

Problems and methods

In this paper, we focus on using pruning to reduce the size of the KV cache.

Current KV cache pruning methods, such as H_2O (Zhang et al. 2023) and Scissorhands (Liu et al. 2023), utilize attention matrices to compute the importance of each key. Based on this importance, they prune the key-value pairs. However, in their approaches, there is a bias in the statistics as they simply sum the attention matrix to determine the importance of each key. This strategy introduces a statistical bias where earlier queries receive more cumulative scores compared to newer queries. This bias can have a substantial impact on the performance of the model. Furthermore, during the score accumulation process, earlier query scores dominate, causing the model to prioritize global information over local information. As a consequence, specific details may receive insufficient attention, while computational overhead increases.

Motivated by the above observations, we propose modified attention score matrix to correct statistical bias and employ a forgetting factor to concentrate on local properties. Specifically, concerning the adjustment of the attention score matrix, as illustrated in section 3, with an increasing number of queries, each individual score should decrease proportionally, while their cumulative sum remains fixed at 1. Through this, we successfully mitigate the statistical bias that can occur during the estimation of importance scores. As for the forgetting factor, we apply a window mechanism, wherein only the last few rows of attention scores are considered for accumulation. By this, we can effectively incorporate local properties while minimizing computational overhead.We have also introduced a novel quantization method that dynamically adjusts the quantization precision based on the correlation between the key (K) and value (V) vectors with the query (Q). Keys highly correlated with the query are quantized with high precision, while less relevant or irrelevant keys are compressed with lower resolution. This flexible quantization process is expected to enhance the efficiency of the KV cache in large models.

Problem Description and Related Work

This paper explore the memory implications of the LLM attention procedure, particularly emphasizing the storage aspects of the key-value pairs in attention. Assume a model with hidden dimension represented as d. Let b be the batch size, and s indicate the length of input prompt sentences.

For the model parameters, we use $W_K^i \in R^{d\times d}$ and $W_V^i \in R^{d\times d}$ as the key and value projection matrices for the i^{th} layer, respectively.

The LLM attention process involves an initial phase where the model accepts the prompt sentences. The key and value embeddings in the attention mechanism are cached to minimize repeated computations. For the input at the i^{th} transformer layer, we can express it as: $X_{prompt} = [x^1_{prompt}, \ldots, x^s_{prompt}] \in R^{b\times s\times d}$. The corresponding key and value caches are given by $K^i = X_{prompt}W_K^i$ and $V^i = X_{prompt}W_V^i$ As the model moves to the generation phase, it utilizes the cached key-value pairs from the prompting phase and produces one token at each step. For every iteration t in the i^{th} transformer layer, the attention input modifies the cache as follows: $x_t^i \in R^{b\times 1\times d}$, $K_{t+1}^i = [K_t^i, x_t^i W_K^i]$, $V_{t+1}^i = [V_t^i, x_t^i W_V^i]$ (Liu et al. 2023).

LLM Inference Memory Breakdown

In this section, we detail the memory usage distribution for LLMs. The memory allocation is divided into three components: model weights, KV cache, and the activation buffer (Liu et al. 2023). The model weights' size varies based on model settings like the count of transformer layers and the hidden dimension. The KV cache's size is influenced by model parameters, sequence duration, and batch volume. Meanwhile, the activation buffer's size is determined by parallel tactics, model specifications, and the chosen implementation. Notably, the activation buffer occupies significantly less memory compared to the other two components.

Efficient Attention

Attention mechanisms are at the core of LLMs, determining how the model allocates attention and processes information. Therefore, the design and implementation of efficient attention mechanisms are crucial for enhancing inference efficiency.

Techniques such as quantization, pruning, and distillation are used to reduce the size and complexity of models, thereby lowering memory consumption and computational overhead. Quantization converts model weights from floating-point numbers to lower-precision integer representations (Han, Mao, and Dally 2015). Pruning or sparsity reduces model size by removing redundant components (Molchanov et al. 2016). Distillation trains a smaller model to mimic the behavior of a larger model, reducing memory

consumption and computational overhead (Hinton, Vinyals, and Dean 2015). The methods discussed in this paper are closely linked to sparsity, but primarily address the unique challenge of KV cache in inference.

The quadratic complexity of attention modules is a key constraint in transformer inference, as highlighted in (**?**). Several strategies, including Reformer (Kitaev, Kaiser, and Levskaya 2020), Performer (Choromanski et al. 2020), Sparse Transformer (Child et al. 2019) and H_2O (Zhang et al. 2023) have emerged to tackle this issue. In this paper, we use the most relevant studies that develop algorithms to reduce the memory footprint of the KV cache as our baseline references.

Quantization and Sparsification

Quantization(Krishnamoorthi) in Large Language Models (LLMs) like ChatGPT is a process aimed at reducing the precision of weights and activations in neural networks, enhancing computational efficiency and reducing memory usage. This is especially vital for deploying models in environments with limited resources, such as mobile devices or edge computing platforms. Quantization Aware Training (QAT) and Post Training Quantization (PTQ) are two primary methods used. QAT integrates quantization operations into the neural network before training, but is less suitable for LLMs due to the high costs and complexity of training these models. PTQ, on the other hand, involves quantizing the weights or both weights and activations after training, without further training(Yao et al. 2022). In LLMs, quantizing both weights and activations simultaneously can lead to significant performance degradation, making weights-only quantization a more common approach.

Quantizing the Key-Value (KV) cache in LLMs, however, presents unique challenges. In text generation applications, key-value embeddings are calculated once and cached for use in all subsequent queries. While a method similar to weights-only quantization could be applied to the KV cache, the dynamic nature of key-value embeddings and the variability in output sequence lengths complicates the process. The key difference lies in the fact that the information needed for quantizing weights is inherent in the weights themselves, whereas key-value embeddings are dynamically calculated. This dynamic aspect makes it difficult to determine an appropriate dimension for splitting the KV cache and to find suitable example inputs for fine-grained group-wise quantization calibration offline, areas which are less explored and require innovative solutions.

Sparsification is another way to compressing the KV cache besides quantization. Scissorhands(Liu et al. 2023)find that the attention maps with high scores are repeated. They compresses the KV cache by only saving the key-value embeddings with high attention scores. Optimizing the model structure can also compress the KV cache. Shazeer (Shazeer 2019)uses the multi-query attention to replace the multi-head attention to reduce the memory consumption of the KV cache.

In our research, we make three significant contributions. Firstly, we correct the bias in attention statistics. Secondly, we introduce a forgetting factor to exclude the influence of early queries on keys. Thirdly, We combine Quantization and Sparsification, and perform tiered quantization on Keys and Values based on their scores. Lastly, our experiments demonstrate excellent performance improvements in memory-efficient LLM inference.

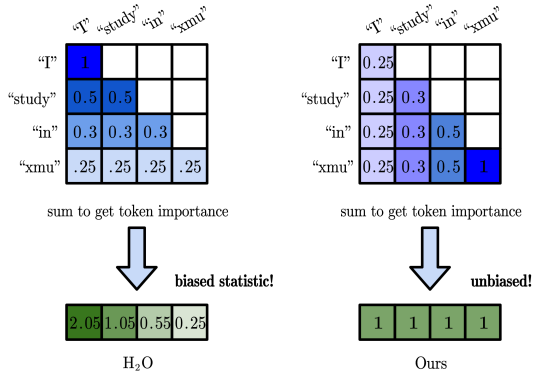

Figure 2: Our method for correcting statistical bias involves multiplying each element of the Attention matrix by a factor $i/(n-j+1)$ to obtain the adjusted version. The adjusted version ensures that the expected importance of each key, after summation, is equal to 1.

Method

Similar to $H2O$ (Zhang et al. 2023), we will employ attention scores as a metric to evaluate the significance of each key. Our plan is to apply transformations to the attention scores to mitigate any statistical bias. Moreover, we will discuss the importance of incorporating a forgetting factor into the process.

Correcting Statistical Bias

In the task of next token prediction, a query at a specific position can only access the preceding keys. The i-th query can only see the first i keys and calculate the dot product with them, resulting in i attention scores. However, if we sum the attention matrix column-wise, it will cause early queries to accumulate a larger number of scores compared to new queries. This leads to a statistical bias that can significantly impact the model's performance. The model tends to favor retaining keys located earlier in the sequence because they have accumulated higher scores.

Moreover, it is crucial to note that when conducting attention calculations between the i-th query and the preceding i keys, we obtain i scores, and their summation amounts to 1. As a result, as i grows larger, each individual score diminishes proportionally. To illustrate, let's consider the case where $i = 1$, indicating the presence of just one token, resulting in a score of 1. Nonetheless, as i increases, the scores progressively decrease, ensuring that their cumulative sum remains constant at 1.

Therefore, we propose a modification where we divide the j-th column of the attention score matrix by the factor $n -$

$j+1$ and multiply the i-th row by the factor i, resulting in

$$Score_{i,j} \leftarrow \frac{i}{n-j+1} \times Score_{i,j} \tag{1}$$

The division of the j-th column by $n-j+1$ is because the j-th column accumulates a total of $n-j+1$ scores. We aim to mitigate the impact of different numbers of accumulated scores for keys at different positions on statistical importance.

And the multiplication of the i-th row by i is carried out to ensure that the expected score for each key, as observed by the query, is 1.

$$i \times \mathbb{E}[Score_{i,:}] = i \times \left[\frac{1}{i}, \frac{1}{i}, ..., \frac{1}{i}\right] \tag{2}$$

After the transformation, we derive new scores, and by summing them column-wise, the i-th element of the resulting vector represents the importance score of the i-th key. These scores have an expected value of 1 for each key. Consequently, through this transformation, we successfully mitigate the statistical bias that can occur during the estimation of importance scores.

Forgetting Factor

Text often demonstrates local dependencies, indicating that the influence between two tokens diminishes as their distance increases. Hence, during the score accumulation process, it is undesirable to include early query scores for each key. Instead, we aim to focus solely on the scores from recent queries in order to capture the most relevant information. This concept can be realized by applying a window mechanism, wherein only the last few rows of attention scores are considered for accumulation. By adopting this approach, we can effectively incorporate local properties while minimizing computational overhead.

Quantization Dropout

In the traditional attention mechanism of neural networks, particularly transformer models, attention scores are computed using the dot product of Query (Q) and Key (K) matrices. These scores reflect the relevance of each element in the input sequence, where higher scores denote stronger relationships or greater relevance between the elements represented by the query and a particular key. After computing these scores, they are normalized using a softmax function, which converts them into probabilities. These probabilities are then used to weight the Value (V) vectors, aggregating information from V based on the relevance indicated by the QK scores. This process is central to determining which parts of the input data the model should focus on. Due to the normalization of Softmax, the quantization error of the K cache has less influence on the output. (Zhao et al. 2023)

The concept of quantizing Key (K) and Value (V) vectors based on their calculated relevance to the Query (Q) is an innovative approach to optimize memory usage in neural networks, as 3 shows. In this method, the most relevant keys would be quantized with higher precision (e.g. 8 bits), while less relevant or irrelevant keys would be compressed using lower resolutions (e.g. 4bits). This implies a dynamic quantization process, where the degree of quantization is determined by the attention scores. Similarly, this principle could be extended to the V vectors, where the relevance of each value, as derived from the QK calculations, would dictate the level of precision in quantization. Such a method suggests a more memory-efficient approach by dynamically allocating different precision levels based on the relevance of each key and value in the attention process, potentially enhancing the efficiency of KV cache in large models.

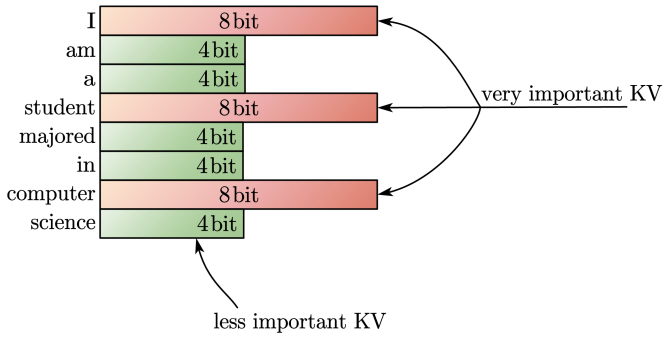

Figure 3: The mechanism of KV cache quantization dropout.

Experiments

First, we will demonstrate how our approach can reduce the KV cache size by up to 80% while maintaining model capacity in llama-13B and llama-7B (Touvron et al. 2023). Next, we will present ablation experiments to validate the effectiveness of correcting biased statistics, the forgetting factor, and quantization dropout.

End-to-End Results

In this section, we will use Huggingface's Llama-13B and Llama-7B models as baseline models. Building upon these models, we will conduct experiments involving bias correction statistics, forgetting factor, and quantization dropout.

For our testing data platform, we will utilize LM Eval Harness(Gao et al. 2021). This platform will be used to evaluate the performance on datasets such as OpenbookQA(Mihaylov et al. 2018), Winogrande(Sakaguchi et al. 2021), and Wikitext2(Merity et al. 2016). We will measure the accuracy of the models for OpenbookQA and Winogrande, while for Wikitext2, we will assess the perplexity.

The OpenbookQA and Winogrande datasets will be evaluated based on their accuracy, indicating how well the models perform in answering questions. On the other hand, the Wikitext2 dataset will be assessed using perplexity, which measures the quality of language modeling by quantifying how well the model predicts the next word in a sequence.

By incorporating bias correction statistics, forgetting factor, and quantization dropout techniques, we aim to enhance the performance of the Llama-13B and Llama-7B models on these evaluation tasks.

For quantization dropout, we adopted a scheme that combines high-precision bits with low-precision bits. The KV

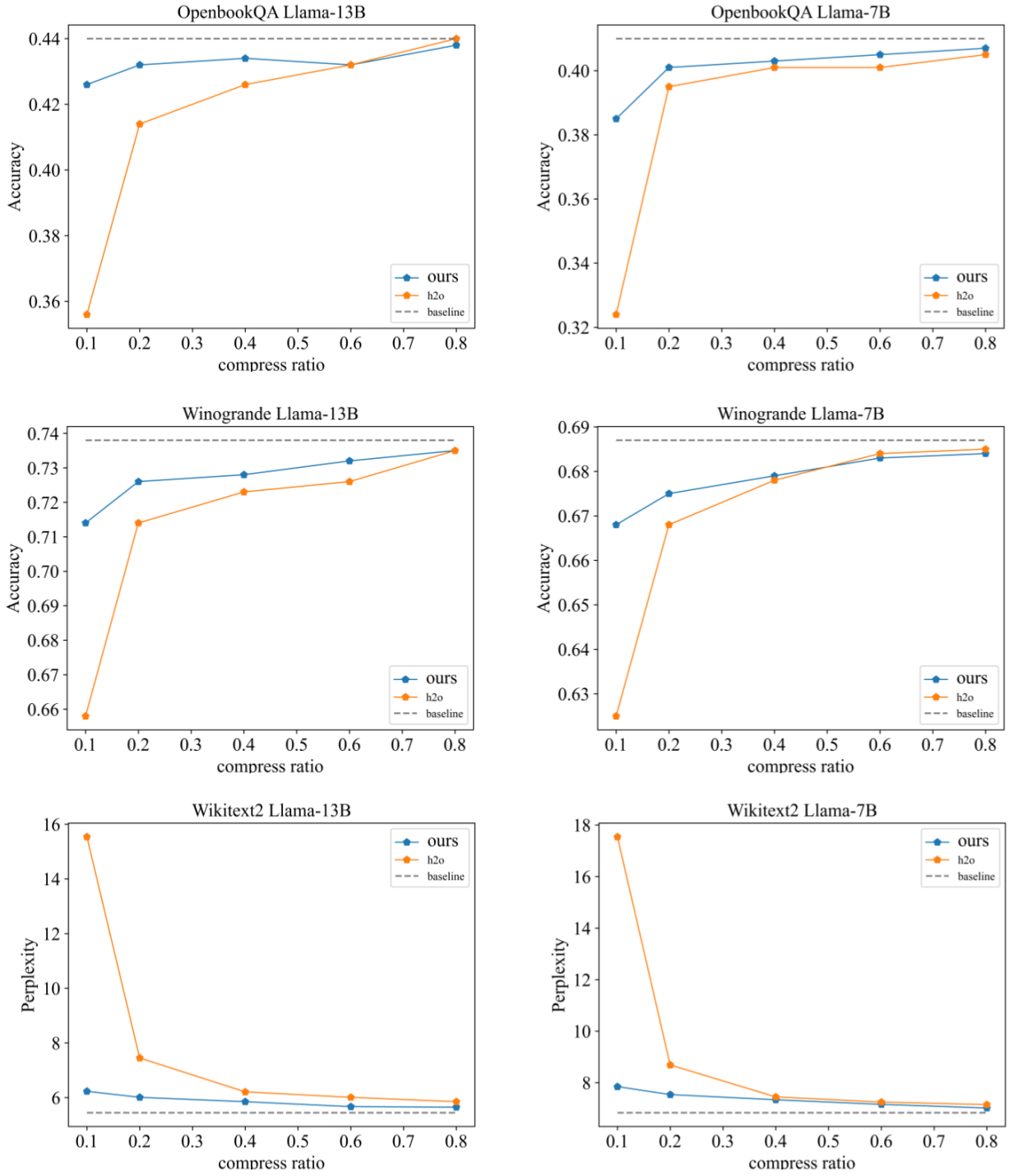

Figure 4: We conducted accuracy tests on OpenbookQA and Winogrande, as well as perplexity tests on Wikitext2, using the Llama-13B and Llama-7B models. The results are represented by the blue line for our approach and the orange line for token pruning methods, with H2O as a representative.

Compress Ratio	Quantization Schemes
0.1	20% 4bit + 80% 1bit
0.2	60% 4bit + 40% 2bit
0.4	60% 8bit + 40% 4bit
0.6	80% 8bit + 20% 4bit
0.8	60% 16bit + 40% 8bit

Table 1: The quantization schemes used for different compression rates.

cache of important tokens was quantized with high precision, while the KV cache of non-important tokens was quantized with low precision. The compression rate was controlled by adjusting the ratio between high-precision and low-precision bits. 1 illustrates the specific quantization schemes for each compression rate.

As shown in Figure 4, our approach can maintain performance when the compression rate reaches around 0.1 across all datasets. In contrast, H2O exhibits a significant decline in performance at a compression rate of 0.1.

Ablation Results

In this section, we will conduct separate tests to evaluate the effectiveness of statistical bias correction and quantization dropout. This will allow us to clearly and comprehensively assess their impact and performance.

Correcting Statistical Bias Ablation Results In this section, we employ the Llama-13B model as our baseline, along with all our proposed methods. We evaluate the effectiveness of correcting statistical bias by incorporating a forgetting factor. We conduct these evaluations using the OpenbookQA and Wikitext2 datasets. The goal is to assess the impact and effectiveness of our approach in mitigating statistical bias in the models' predictions.

In the ablation experiment results, we can observe that as the compression ratio decreases, the advantages of employing statistical bias correction become more apparent. It plays a crucial role in capturing which tokens are more important and which ones are less important. In comparison, methods like H2O utilize a local budget approach, where a certain number of recent tokens are always retained. On the other hand, our quantization dropout method achieves excellent results without relying on such local budget tricks.

Quantization Dropout Ablation Results In this section, we continue to use the Llama-13B model as our baseline with all our proposed methods. The "w.o." model, on the other hand, excludes the quantization dropout technique. Instead, it retains all tokens within the compression ratio with their original precision, while pruning off non-important tokens outside the compression ratio. Essentially, the "w.o." model is an extension of the H2O approach, incorporating statistical bias correction and a forgetting factor.

From the experimental results, we can observe that as the compression ratio gradually decreases, there is initially minimal decrease in accuracy compared to the baseline model. It is only when the compression ratio reaches 0.2 or below that the quantization dropout method starts to demonstrate its effectiveness. This implies that when the compression ratio becomes too low, it is no longer possible to retain all the crucial tokens, resulting in a decline in accuracy. However, the proposed method in this paper addresses this issue through soft pruning, ensuring that all tokens are preserved. The key tokens are retained with high precision, while non-key tokens are retained with lower precision.

Thus, it is evident that the quantization dropout method plays a crucial role in further reducing the compression ratio.

Summary

This paper addresses the issue of memory consumption in key-value (KV) caches during language model inference. A pruning approach is proposed to reduce the KV cache size by evaluating the importance of keys using attention scores. Statistical bias is corrected, a forgetting factor is introduced, and an innovative quantization method is adopted. The method is demonstrated in experiments to reduce the KV cache size by up to 80% in llama-13B and llama-7B models while maintaining model capacity. Performance evaluation using LM Eval Harness validates the effectiveness on datasets such as OpenbookQA, Winogrande, and Wikitext2. The introduced quantization dropout method dynamically adjusts precision based on relevance, maintaining performance and improving KV cache efficiency. Ablation experiment results highlight the impact of statistical bias correction on model predictions. The quantization dropout method effectively preserves model accuracy during compression ratios reduction through soft pruning. Overall, these improvements aim to optimize KV cache memory utilization, enhancing the overall performance of language models.

References

Child, R.; Gray, S.; Radford, A.; and Sutskever, I. 2019. Generating Long Sequences with Sparse Transformers. *arXiv: Learning,arXiv: Learning*.

Choromanski, K.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J.; Mohiuddin, A.; Kaiser, L.; Belanger, D.; Colwell, L.; and Weller, A. 2020. Rethinking Attention with Performers. *arXiv: Learning,arXiv: Learning*.

Gao, L.; Tow, J.; Biderman, S.; Black, S.; DiPofi, A.; Foster, C.; Golding, L.; Hsu, J.; McDonell, K.; Muennighoff, N.; et al. 2021. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*.

Han, S.; Mao, H.; and Dally, W. 2015. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv: Computer Vision and Pattern Recognition,arXiv: Computer Vision and Pattern Recognition*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *arXiv: Machine Learning,arXiv: Machine Learning*.

Kitaev, N.; Kaiser, ; and Levskaya, A. 2020. Reformer: The Efficient Transformer. *arXiv: Learning,arXiv: Learning*.

Compress Ratio	OpenbookQA			Wikitext2		
	baseline	w.o. correct	w.o. qdrop	baseline	w.o. correct	w.o. qdrop
0.1	0.428	0.377	0.324	6.32	9.24	17.67
0.2	0.434	0.412	0.428	6.05	7.31	6.84
0.4	0.436	0.425	0.432	5.96	6.14	6.12
0.6	0.432	0.430	0.435	5.74	5.94	5.68
0.8	0.438	0.434	0.438	5.67	5.76	5.53

Table 2: Ablation experiments results. **w.o. correct** means without correcting statistical bias and forgetting factor, and **w.o qdrop** means without quantization dropout.

Krishnamoorthi, R. ???? Quantizing deep convolutional networks for efficient inference: A whitepaper.

Liu, Z.; Desai, A.; Liao, F.; Wang, W.; Xie, V.; Xu, Z.; Kyrillidis, A.; and Shrivastava, A. 2023. Scissorhands: Exploiting the Persistence of Importance Hypothesis for LLM KV Cache Compression at Test Time. *arXiv preprint arXiv:2305.17118*.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2016. Pruning Convolutional Neural Networks for Resource Efficient Inference. *Cornell University - arXiv,Cornell University - arXiv*.

Sakaguchi, K.; Bras, R. L.; Bhagavatula, C.; and Choi, Y. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9): 99–106.

Shazeer, N. 2019. Fast Transformer Decoding: One Write-Head is All You Need. *arXiv: Neural and Evolutionary Computing,arXiv: Neural and Evolutionary Computing*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

Yao, Z.; Aminabadi, R.; Zhang, M.; Wu, X.; Li, C.; and He, Y. 2022. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers.

Zhang, Z.; Sheng, Y.; Zhou, T.; Chen, T.; Zheng, L.; Cai, R.; Song, Z.; Tian, Y.; Re, C.; Barrett, C.; Wang, Z.; and Chen, B. 2023. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models.

Zhao, Y.; Lin, C.-Y.; Zhu, K.; Ye, Z.; Chen, L.; Zheng, S.; Ceze, L.; Krishnamurthy, A.; Chen, T.; and Kasikci, B. 2023. Atom: Low-bit Quantization for Efficient and Accurate LLM Serving. *arXiv preprint arXiv:2310.19102*.