Q-HITTER: A BETTER TOKEN ORACLE FOR EFFICIENT LLM INFERENCE VIA SPARSE-QUANTIZED KV CACHE

Zhenyu Zhang [*1] **Shiwei Liu** [*2 3] **Runjin Chen** [1] **Bhavya Kailkhura** [4] **Beidi Chen** [5 6] **Zhangyang Wang** [1]

ABSTRACT

This paper focuses on addressing the substantial memory footprints and bandwidth costs associated with the deployment of Large Language Models (LLMs). LLMs, characterized by their extensive context length (e.g., $\geq$4096), inherently demands vast memory resource and traffic to store and load the attention key and value embeddings within self-attention modules, referred to as the KV cache. In an effort to alleviate these resource-intensive aspects of LLM inference, techniques such as sparsification and quantization for KV cache reduction have been investigated as separate endeavors within the realm of LLMs. However, this paper illuminates the critical importance of considering the compound effects of these techniques when employed together, as a simplistic amalgamation of sparsification and quantization can yield sub-optimal performance. For instance, the "Heavy Hitter Oracle (H_2O)" (Zhang et al., 2023b) has demonstrated that preserving just 20% of the KV cache attributed to pivotal tokens, denoted as "Heavy Hitters", can yield substantial memory savings while upholding the model's original performance. Furthermore, the KV cache of these "Heavy Hitter" tokens, which are identified as those with the highest accumulated attention scores, can be further quantized with encouraging throughput saving. Nevertheless, our investigation uncovers two primary deficiencies in such unrefined post-sparsification quantization in low-bit scenarios: (1) the application of low-bit KV cache quantization, specifically $\leq$ 4-bit, significantly diminishes the accuracy of Heavy Hitter selection during the generation phase, particularly in deeper layers; (2) tokens selected by the "Heavy Hitter Oracle" are not necessarily well-suited for quantization, and their quantization can lead to sub-optimal performance. To overcome these challenges, we propose a novel rule-of-thumb for token selection during LLM generation, termed **Q-Hitter**. This approach combines both accumulated attention scores and "Quantization Friendliness" metrics for different layers, identifying tokens that are not only pivotal for preserving the generalization capabilities of LLMs but are also more amenable to KV cache quantization. Q-Hitter naturally offers a free lunch of KV cache quantization and can further escalate the affordability of state-of-the-art LLMs. Additionally, we also demonstrate that Q-Hitter empowers LLMs to effectively handle inputs of infinite sequence length, enhancing the capacity of LLMs to process a more extensive range of informations. Extensive experiments conducted across various LLMs and tasks substantiate the superiority of the proposed Q-Hitter framework over the original H_2O framework. Remarkably, Q-Hitter achieves full model quality preservation while delivering up to a remarkable **20**$\times$ reduction in memory usage and up to **33**$\times$, **7**$\times$, **4**$\times$ and **1.3**$\times$ throughput improvements compared with the Hugginface Accelerate, DeepSpeed, FlexGen and H_2O, respectively. Codes are available at https://github.com/VITA-Group/Q-Hitter.

1 INTRODUCTION

Large language models (LLMs) are show-stealers in modern-day natural language processing applications. As we consistently push the boundaries of model size and expand their training datasets, LLMs begin to showcase outstanding performance across a wide spectrum of tasks, as documented in recent studies (Brown et al., 2020; Thoppilan et al., 2022; Chowdhery et al., 2022; Chiang et al., 2023) with unexpected emerging behaviors (Wei et al., 2022; Srivastava et al., 2022).

While LLMs have been demonstrating increasingly remarkable performance, deploying these colossal models on commodity hardware is a daunting challenge, not only due to their large amount of parameters but also the massive KV cache memory footprints when performing text generation with long attention contexts. For example, with a batch size of 512 and a context length of 2048, the KV cache of

[*]Equal contribution [1]University of Texas at Austin [2]University of Oxford [3]Eindhoven University of Technology [4]Lawrence Livermore National Laboratory [5]Carnegie Mellon University [6]Meta AI (FAIR). Correspondence to: Zhangyang Wang <atlaswang@utexas.edu>.

Proceedings of the 5^{th} MLSys Conference, Santa Clara, CA, USA, 2024.

a 500B+ model with multi-head attention accumulates to 3TB in total, which will surpass the model's parameter size by three times (Pope et al., 2023). Meanwhile, there has been a strong prevailingness recently to train LLMs with longer context windows, which can enhance the capacity of LLMs to effectively process a more extensive range of information, leading to better performance when supporting extended conversation histories in chat applications, tackling diverse summarization tasks, and comprehending longer documents (Touvron et al., 2023). Therefore, finding ways to reduce KV cache memory footprints is pivotal to improving the efficiency and performance of LLMs.

Recently, Zhang et al. (2023b) observed that a small portion of tokens (dubbed as *Heavy-Hitters*) contributes most of the value when computing attention scores. Based on this observation, they proposed Heavy-Hitter Oracle (H_2O), a low-cost framework that achieves a significant reduction of KV cache memory footprints by dynamically maintaining only 20% KV cache of these Heavy-Hitters. At each generation step, Heavy-Hitters tokens are dynamically selected by identifying the top-K tokens with the highest accumulated attention scores. Furthermore, H_2O demonstrates appealing compatibility with quantization, achieving throughput improvement when quantizing the sparsified KV cache into low bit. While demonstrating good compatibility with quantization, we argue that directly applying quantization on the top of KV cache sparsification without carefully considering their interaction can easily lose critical information, resulting in undesirable damage to LLM's generalization ability. This issue is expected to be amplified for the low-bit quantization scenario where the limited precision is insufficient to represent complex relationships in the data.

Our preliminary exploration verifies our concerns and delivers very intriguing phenomena, which are summarized as the following:

#1. The critical Heavy Hitter tokens identified by H_2O in the generation phase exhibit a notable shift after quantization. Remarkably, post-4-bit quantization, more than 50% of the originally deemed important tokens are reclassified as having minimal utility. This shift in attention becomes increasingly conspicuous in deeper layers. Such attention shift serves as the root cause of the performance loss in low-bit quantization scenarios where LLMs suffer from notable perplexity increase, as shown in Section 4.2.

#2. Furthermore, it is imperative to acknowledge that not all tokens designated by H_2O are amenable to quantization, ultimately resulting in suboptimal performance after quantization. In our empirical investigation, expounded in Section 4.3, we find that merely the top 1% of tokens preserved by the "Heavy Hitter Oracle" exhibit quantization-friendly attributes, which is way smaller than the least amount of tokens required by H_2O, i.e., 20%. Consequently, it becomes important to identify tokens that fulfill a dual role: preserving essential generalization capabilities while also being amenable to the quantization process.

In order to overcome these hurdles, this paper introduces a new token selection framework, **Q-Hitter**, for KV cache compression associated with an additional metric named as "Quantization Friendliness". Combining this metric with the accumulated attention scores, Q-Hitter is designed to pinpoint tokens that play a crucial role in preserving the generalization abilities of LLMs while also being particularly well-suited for the quantization of KV cache, providing a natural bonus of KV cache quantization for free. With extensive experiments across a range of LLMs and tasks, we demonstrate that our approach consistently improves the quantization performance of H_2O, achieving near lossless performance with up to $20\times$ memory saving and $33\times$ throughput improvement.

Our key contributions are summarized as follows:

- We first investigate into the naive combination of sparsification and quantization for KV cache compression. Two main hurdles exhibit: (1) utilizing low-bit quantization for KV cache, specifically when employing $\leq$ 3 bits, leads to a notable reduction in the accuracy of the Heavy-Hitter selection during the generation phase, particularly within deeper layers. (2) The tokens identified by the H_2O do not consistently exhibit compatibility with quantization, and the quantization of these tokens often results in less-than-optimal performance.
- To tackle these issues, we introduce Quantization-aware Heavy-Hitter Oracle (Q-Hitter), a rule-of-thumb for identifying crucial tokens that are also highly quantization-compatible within KV cache. Q-Hitter seamlessly incorporates KV cache quantization, achieving lossless 4-bit quantization on complex tasks like text summarization and multi-document question answering, all while drastically cutting down memory needs for efficient LLM deployment.
- Additionally, Q-Hitter empowers LLMs to tackle infinite-length input sequences, a notable challenge for current LLMs, by dynamically preserving significant and quantization-friendly tokens as attention sinks. Combined with the position rolling, Q-Hitter allows for effective token generation up to 4 million in length.
- Extensive experiments validate the effectiveness of our methods. With only a 5% to 12.5% memory budget, Q-Hitter maintains the full performance of the original LLMs across diverse tasks. Notably, it showcases throughput enhancements of $33\times$, $7\times$, $4\times$, and $1.3\times$ compared with Accelerate, DeepSpeed, FlexGen, and H_2O, respectively.

2 Related Work

LLMs have a large memory footprint both due to the massive model parameters and the transient state needed during decoding (Dettmers et al., 2023b). In this section, we will briefly summary how advanced techniques are utilized to address these challenges.

Pruning and Quantization. Model compression is a long-standing group of techniques with the primary goal to reduce model size with negligible performance damage. These techniques can be loosely categorized into two primary groups, Pruning and Quantization. ❶ Pruning or sparsity (LeCun et al., 1989; Han et al., 2015) is able to eliminate superfluous components in neural networks with negligible performance loss. In the context of LLMs, one-shot pruning without re-training is more appealing due to the huge expenses of the re-training (Jaiswal et al., 2023; Sun et al., 2023; Frantar & Alistarh, 2023; Ma et al., 2023). ❷ Quantization is used to reduce the memory and computational requirements by representing the model's parameters, such as weights and activations, with lower bit precision (Dettmers et al., 2023a; Frantar et al., 2022; Cheng et al., 2023; Li et al., 2023a; Behdin et al., 2023; Li et al., 2023b; Chee et al., 2023; Wu et al., 2023; Liu et al., 2023b;c; Zhang et al., 2023a). A primary challenge in quantized LLMs is the emergence of outliers (activations with large magnitude (Xiao et al., 2023a) or weights associated with large quantization loss (Lin et al., 2023)) exhibited in LLMs. Numerous approaches have been proposed to overcome this issue, including rescaling techniques (Xiao et al., 2023a; Lin et al., 2023; Wei et al., 2023), clustering strategies (Yuan et al., 2023; Kim et al., 2023), and mixed-precision methods (Dettmers et al., 2023b; Lee et al., 2023). While these approaches have demonstrated promising efficacy in model compression, their primary goal is to handle barriers caused by the massive parameters and activations, overlooking the KV cache, which typically results in significant memory usage in tasks that involve long-context generation.

Attention Approximation. The quadratic computational complexity of attention mechanisms stands as a significant impediment to LLM inference, prompting the development of numerous efficient self-attention variants aimed at reducing self-attention complexity. Notably, BigBird (Zaheer et al., 2020) extends this by advocating for attention to a random subset of prior tokens and a specific set of globally accessible tokens. Linformer (Wang et al., 2020) leverages a low-rank approximation of the attention matrix, while Performer (Choromanski et al., 2020) employs a non-softmax kernel for enhanced efficiency. Longformer (Beltagy et al., 2020), on the other hand, implements dilated sliding window patterns to expand the attention's receptive field, assigning distinct window sizes to each layer manually. Flash Attention (Dao et al., 2022) addresses the quadratic memory demands of attention mechanisms. Additionally, variants like Sparse Transformer (Child et al., 2019), low-rank based transformers (Katharopoulos et al., 2020), and multi-query attention models (Pope et al., 2023; Chowdhery et al., 2022; Shazeer, 2019) contribute to cache size reduction, albeit not specifically designed for complexity alleviation. However, these innovations either fail to yield cache reduction benefits (Dao et al., 2022; Kitaev et al., 2020) or induce substantial performance declines when directly applied to pre-trained LLMs for generation tasks (Katharopoulos et al., 2020; Pope et al., 2023; Chowdhery et al., 2022).

KV Cache Reduction. Caching, quintessential for enhancing system performance, necessitates the formulation of proficient eviction strategies to manage data that is frequently accessed. Traditional methodologies, including Least Recently Used (LRU) and Least Frequently Used (LFU) (O'neil et al., 1993; Lee et al., 2001), focus on optimizing data access based on its recency and frequency, respectively. The design of KV cache, grapples with numerous challenges akin to those faced in conventional caching paradigms. Some recent works demonstrate quite promising progress in mitigating memory requirements of KV cache for better inference efficiency. SpAtten (Wang et al., 2021) implements cascade pruning during training to eliminate inconsequential tokens and attention heads on the fly. Landmark Attention (Mohtashami & Jaggi, 2023) introduces landmark tokens to mark distinct input blocks, steering attention towards relevant blocks. Similarly, (Mu et al., 2023) leverages gist tokens to encapsulate key information, thereby streamlining the KV cache, albeit at an additional training expense. CoLT5 (Ainslie et al., 2023) employs a routing-based method, diverting specific tokens to lighter attention blocks to alleviate cache cost. LM-Infinite (Han et al., 2023) and StreamLLM (Xiao et al., 2023b) adopt a strategy that retains only a selection of initial and local tokens, reducing cache overhead. While H_2O identifies a subset of tokens as particularly vital for the generation, developing a KV cache eviction policy centered around this insight to achieve impressive performance enhancements. Moreover, FlexGen (Sheng et al., 2023a) applies group-wise quantization to the KV cache, neglecting the variance in quantization difficulty across different tokens. In our work, we approach this challenge from a novel angle, proposing a quantization-aware eviction policy that prioritizes the retention of tokens crucial for generation performance and amenable to quantization.

3 Preliminaries

In this section, we briefly summarize the generative inference and quantization in LLMs, as well as the notations that will be utilized in the following.

Generative Inference. The generative inference process of LLMs unfolds in two primary stages: ❶ The prefilling stage where an input sequence is utilized to generate the key-value embeddings for each token across every transformer layer within LLMs. These embeddings are stored in the KV cache; ❷ The generation stage retrieves embeddings from the cache and generates new tokens sequentially. Every newly generated token then serves as an input for the subsequent generation step and its corresponding key-value embeddings are used to update the KV cache. Given the iterative nature of this stage, the memory requirement for KV cache scales linearly with the number of tokens, possibly leading to prohibitive memory usage that impedes LLM inference. This work targets on accelerating LLM inference via compressing the KV cache during the generation stage.

Quantization. Quantization is a widely used compression technique that transforms high-precision float values into discrete integer values. Since non-uniform quantization requires specialized hardware support that is not broadly available, we focus on uniform quantization. The typical uniform quantization process can be formulated as:

$$\begin{aligned} X_q &= \text{Quant}_n(X, s, z) \\ &= \text{clamp}(\lfloor \frac{X}{s} \rceil + z, -2^{n-1}, 2^{n-1} - 1) \end{aligned} \tag{1}$$

Here, X denotes the original tensor, while X_q represents the quantized tensor. The quantization function Quant_n maps the tensor to an n-bit integer value, utilizing s as the scaling factor and z as the zero point. The clamp function ensures the values are constrained within the n-bit integer range.

Quantization methods fall into two primary categories: static and dynamic quantization. Static quantization calculates the scaling factor s offline, *e.g.*, employing the activations from a calibration dataset to approximate the scaling factor, then applying it to quantize activations. In contrast, dynamic quantization computes the scaling factor in real-time during model execution, which generally causes less performance loss yet requires more computation overhead for quantization.

Furthermore, quantization can be executed at varying granularities. Specifically, per-tensor quantization calculates the scaling factor based on the entire tensor. On the other hand, some finer-grained methods, such as per-channel and per-token quantization, treat each channel and token as separate entities for individual quantization. Also, group-wise quantization divides the tensor into several groups prior to quantization. To select an apt quantization granularity, we examined the dynamic ranges of the Key and Value embeddings, as illustrated in Figure 1. Notably, the activation ranges exhibit significant variance across different tokens, presenting a substantial challenge for quantization. Specifically, some tokens' max values are close to 0 while others are larger than 60. To navigate these disparities and preserve model performance, we use token-wise quantization in the following studies. Meanwhile, token-wise quantization captures the varying quantization difficulty across different tokens, providing a more conducive approach when integrated with token eviction algorithms.

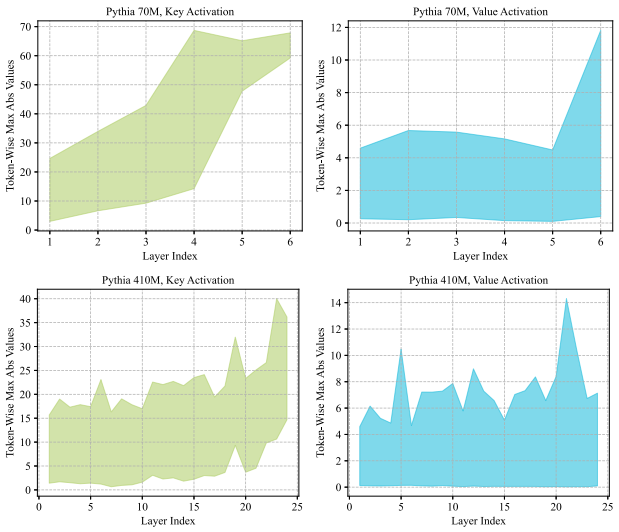

Figure 1. The maximum activation value of Key (left) and Value (right) embeddings on the pre-trained Pythia models. Results are collected on the WikiText-103 validation set.

Notations. We denote query matrix in the attention block as $Q \in \mathbb{R}^{n \times d}$ while the key and value matrix are referred as $K \in \mathbb{R}^{n \times d}$ and $V \in \mathbb{R}^{n \times d}$. The **quantization error** is used to assess the associated difficulty, defined as $E(X) = ||X - \text{Quant}(X)||_2$. Tokens with low quantization errors are referred to as tokens with high **quantization friendiness**. And we use the accumulated attention scores to measure token importance, which is defined as $S_j = \sum_j \text{Softmax}(Q_{j,*}(K_{\leq j,*})^T)$. $Q_{j,*}$ represents the query embedding during the j^{th} decoding step in the generation stage, and $K_{\leq j,*}$ is all the previous key embeddings. The accumulated attention score is an effective metric for identifying the important tokens during the whole generation stage (Wang et al., 2021; Zhang et al., 2023b). We refer to the tokens that exhibit large accumulated attention scores as significant tokens, the same as the concept of Heavy Hitter tokens.

4 METHODOLOGY

This section demonstrates the rationale of the sparsity and quantization co-design method for KV cache compression, followed by the empirical studies of the interplay between sparsity and quantization methods. Then, the design detail of the proposed Q-Hitter framework is described.

4.1 Rationale

The primary goal of this study is to compress the KV cache to enhance throughput during LLM inference. To achieve this, commonly adopted strategies involve exploiting sparsity (i.e., preserving only a crucial subset of KV embeddings (Zhang et al., 2023b; Xiao et al., 2023b)) or employing quantization to reduce memory demands. Nevertheless, the interplay between sparsity and quantization remains ambiguous. Does the introduction of sparsity complicate the quantization process, or does quantization introduce an unacceptable bias when identifying vital KV embeddings? To address these queries, we conduct an empirical investigation to scrutinize the influence of sparsity and quantization.

4.2 H_2O Suffers from Attention Shift after KV cache Quantization

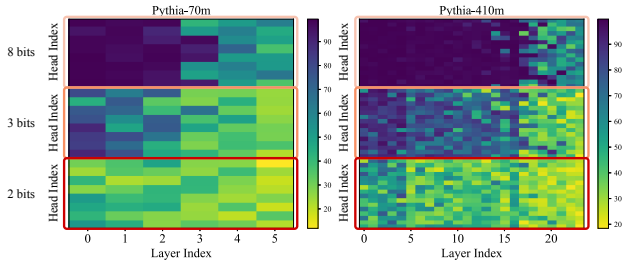

Figure 2. Visualization of the degree of attention shift caused by KV cache quantization, with data derived from WikiText-103 on Pythia-70M and 410M models. The horizontal axis represents various layers, while the vertical axis represents different attention heads. Each block within the matrix displays outcomes associated with different quantization bits. Brighter colors signify a more pronounced impact from quantization on attention shift.

In our initial investigation, we examine the impact of KV cache quantization on the selection of pivotal tokens. Following previous works (Wang et al., 2021; Zhang et al., 2023b), we employ the accumulated attention score to gauge the significance of different tokens. Tokens with higher accumulated attention scores are deemed more crucial in the generation process, and we categorize the top 20% of these tokens into a critical set, $\mathcal{S}$, for inference.

To quantify the attention shift stemming from KV cache quantization, we employ both the original and quantized KV cache to compute accumulated attention scores, subsequently deriving the critical sets $\mathcal{S}_o$ and $\mathcal{S}_q$. Their overlap ratio ($|\mathcal{S}_o \cap \mathcal{S}_q|/|\mathcal{S}_o|$, where $|\mathcal{S}_o| = |\mathcal{S}_q|$) serves as our metric for evaluating attention shifts across various layers and attention heads. The results are showcased in Figure 2. We can observe that: ❶ In terms of selecting significant tokens, quantization causes a minimal effect on the shallow layers, whereas deeper layers exhibit pronounced attention shifts (evidenced by overlap ratios below 50%). ❷ As the quantization bits decrease, the attention shift becomes more prominent, with nearly all layers and heads displaying an overlap ratio below 30% when the KV cache is quantized to ultra-low levels (*i.e.*, 2-bits).

Delving deeper, we analyze the influence of such attention shifts on the effectiveness of KV cache eviction policies. Specifically, we evaluate perplexity on the validation set from WikiText-103 and Penn Treebank datasets while retaining merely a small, crucial subset of KV embeddings. Note that we simulate the quantization process to obtain the shifted subset while preserving the KV embeddings in full precision values. The results are reported in Table 1.

Table 1. Perplexity of language modeling with different set of KV embeddings. "Full" represents using all the KV cache. $\mathcal{S}_o$ stands for the critical subset that is identified by raw accumulated attention scores while $\mathcal{S}_q$ is the associated subset when the accumulated attention scores are shifted by quantization.

Model Size	Dataset	Full	$\mathcal{S}_o$	$\mathcal{S}_q$ 8 bits	$\mathcal{S}_q$ 3 bits	$\mathcal{S}_q$ 2 bits
70M	WikiText-103	55.82	57.61	59.05	63.97	**70.24**
160M		31.57	31.81	31.93	35.15	**37.74**
410M		18.87	18.95	18.95	19.59	**20.18**
70M	Penn Treebank	113.66	117.28	119.69	127.36	**137.91**
160M		68.50	69.51	69.68	74.87	**79.45**
410M		39.02	39.39	39.38	40.81	**42.24**

Without the influence of attention shifts stemming from KV cache quantization, models relying solely on $\mathcal{S}_o$ manage to preserve the quality similar to that of the full model. Nevertheless, as we proceed to lower quantization bits and thus enlarge attention shifts, we observe a marked degradation in performance, reaching up to 24.25 perplexity increase across various model sizes and diverse datasets. The results underscore that KV cache quantization will increase the difficulty of selecting critical tokens and degrade current KV cache eviction algorithms, which motivates us to consider both token eviction and quantization simultaneously.

4.3 Heavy Hitter Tokens Are Not Necessarily Quantization Friendly

Previously, we explored the impact of KV cache quantization on the effectiveness of eviction policy. Now, we turn our focus in the reverse direction, examining how KV cache eviction strategies affect quantization. To start, we conduct an in-depth analysis of token-wise quantization difficulty.

Ultra-crucial tokens are quantization-friendly tokens. To comprehend the interaction between quantization error and the significance of each token in the generation process, we analyze the correlation with accumulated attention scores, visualizing the results in Figure 3. First, we can observe some outlier tokens that have substantially higher accumulated attention scores, underscoring their piv-

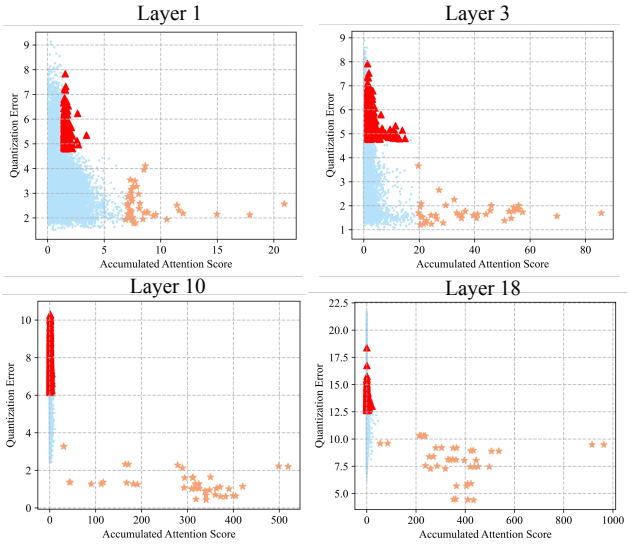

Figure 3. Token-wise distribution of quantization error (vertical axis) and accumulated attention scores (horizontal axis). Results are collected on Pythia-410M with WikiText-2. The tokens with ultra-large accumulated attention scores are highlighted in orange. And the tokens with both top 20% quantization error and accumulated attention scores are marked in red.

otal role during inference. Interestingly, these tokens also demonstrate a propensity for quantization, indicating their quantization-friendly nature. This relationship fortuitously benefits KV cache quantization, providing a natural advantage for reducing memory requirements.

Trade-off between quantization friendiness and token importance. However, relying solely on these ultra-important and quantization-friendly tokens is insufficient for preserving performance. Figure 5 demonstrates the perplexity across various KV cache sizes, illustrating a marked performance degradation when only utilizing outlier tokens (*e.g.*, almost doubling the perplexity compared to full cache performance with Pythia-70M). This indicates a necessary trade-off between quantization-friendliness and significance during LLM inference. As shown in Figure 3, the tokens highlighted in red are characterized by being in the top 20% (*i.e.*, the necessary ratio for preserving performance) for both quantization error and accumulated attention scores. These tokens introduce a new challenge when quantizing the evicted KV cache, motivating the need for a co-design framework that integrates quantization and sparsity to effectively compress the KV cache.

4.4 Q-Hitter

Now, we delve into the details of our Q-Hitter framework. The central idea driving our approach is the simultaneous selection of both significant and quantization-friendly tokens. As shown in Figure 6, Q-Hitter can avoid involving those hard quantized tokens and thus be more robust to quantization. Adopting this strategy offers dual advantages: (i) the accumulated attention score is typically susceptible to the effects of quantization, leading to shifts in attention and affecting the selection of significant tokens. By integrating quantization-friendly tokens, we can mitigate this shift, resulting in a more resilient and robust token eviction policy; (ii) striking a better balance between significance and quantization-friendliness of other tokens ensures that the remaining KV embeddings experience minimal impact from quantization, thereby reducing information loss.

Figure 4 provides an illustration of our Q-Hitter framework, and the specific steps of the algorithm are outlined in Algorithm 1. Throughout the generation process, we record the accumulated attention score and quantization error associated with each token. We subsequently employ a linear combination of these two metrics, moderated by the hyperparameter λ, to strike an optimal balance between the significance and quantization-friendliness of each token. This approach proves particularly beneficial in scenarios requiring extremely low-bit quantization, ensuring the selection of a superior token subset.

Algorithm 1 Q-Hitter Algorithm

Require: A pre-trained LLM $f(\theta, ,)$, KV cache $\mathcal{M}$, token eviction ratio s, quantization bits k, input content X, generation length l, balance factor λ.
Ensure: Generated text Y.
Set $\mathcal{M} = \varnothing$; $Y = \varnothing$; and $i = 0$
while $i < l$ **do**
  # Generation Next Token
  $Q_i, K_i, V_i, y = f(\theta, X, \mathcal{M})$
  # Update KV cache
  $S_s = \sum_i \text{Softmax}(Q_{i,*}(K_{\leq i,*})^T)$
  $S_{qk} = ||K - Quant(K)||_2$
  $S_{qv} = ||V - Quant(V)||_2$
  Normalize S_s, S_{qk}, S_{qv}
  # Linear combination of scores, where $1 - S_{q*}$ measures the quantization friendiness.
  Select tokens with largest S, where $S = \lambda S_s + (1 - \lambda)(1 - S_{qk} + 1 - S_{qv})$, perform quantization on K_i, V_i and update KV cache $\mathcal{M}$
  $y \to Y$; $y \to X$; $i = i + 1$
end while

5 Experiments

In this section, we undertake a comprehensive empirical evaluation to determine the efficacy of our Q-Hitter framework. Our key findings are as follows: (i) **Performance Superiority**: Q-Hitter consistently outperforms the traditional method of applying sparsity and quantization sequentially

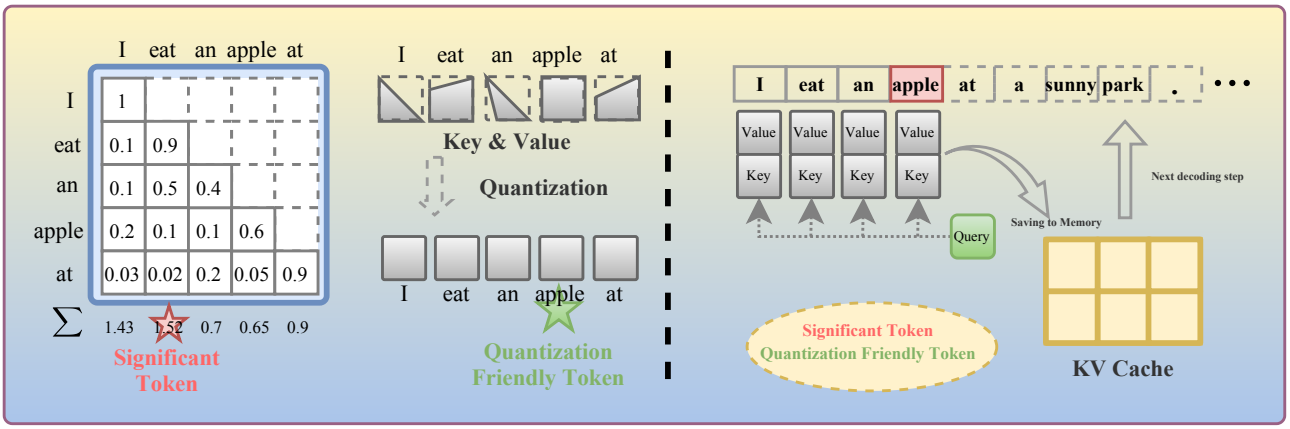

Figure 4. The overall illustration of our Q-Hitter framework where the left part visualizes the significant and quantization-friendly tokens and the right part details the generation process.

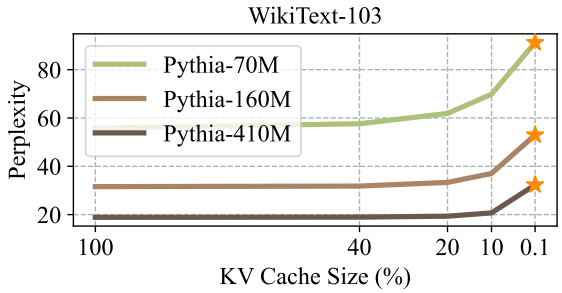

Figure 5. Performance when reducing the size of KV cache via token eviction. The results of only storing the outlier KV is highlighted as "star".

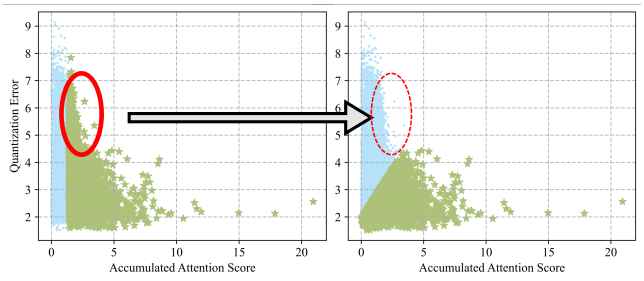

Figure 6. Illustration of token selection results of Q-Hitter (right) and H_2O (left). The green dots represent the selected tokens, and the red circle shows the hard quantized tokens that have been selected by H_2O.

for KV cache compression. Remarkably, it manages to retain the performance equivalent to that of the full model, even under a substantial compression ratio of up to 20×. (ii) **Throughput Enhancements**: Building on the top of Flexgen, Q-Hitter showcases impressive throughput gains, achieving up to 33× compared to other leading systems. (iii) **Handling Infinite Input Sequences**: Our Q-Hitter algorithm empowers LLMs with the capability to process inputs of infinite sequence length.

5.1 General Setup

Models and Datasets. In our experiments, we consider four representative families of Large Language Models (LLMs), including OPT (Zhang et al., 2022), Pythia (Biderman et al., 2023), Llama-2 (Touvron et al., 2023), and Vicuna (Chiang et al., 2023). We have chosen four tasks to benchmark the performance of our Q-Hitter framework, categorizing them into three main groups: ❶ **Text Summarization**: We select two tasks from the popular HELM (Liang et al., 2022) framework, specifically XSUM (Narayan et al., 2018) and CNN/Daily Mail (Nallapati et al., 2016). The performance for these tasks is reported using the Rouge-L score. ❷ **Multi-Document Question Answering (MDQA)**: Following the procedure outlined in (Liu et al., 2023a), we evaluate our method on MDQA tasks. Each test sample consists of one question accompanied by ten documents. The relevant information for the question is contained within one of the documents. By changing the order of the documents, we assess the LLMs' understanding ability when the pertinent information is presented in different positions. The performance is measured using the EM score. ❸ **Language Modeling**: We concatenate all articles in the test set of PG-19 (Rae et al., 2019), which results in overall 4 million tokens and shows Q-Hitter can tackle infinite-long length inputs. Through this diverse set of tasks, we aim to provide a comprehensive evaluation of the effectiveness of different KV cache compression strategies across various model architectures and applications.

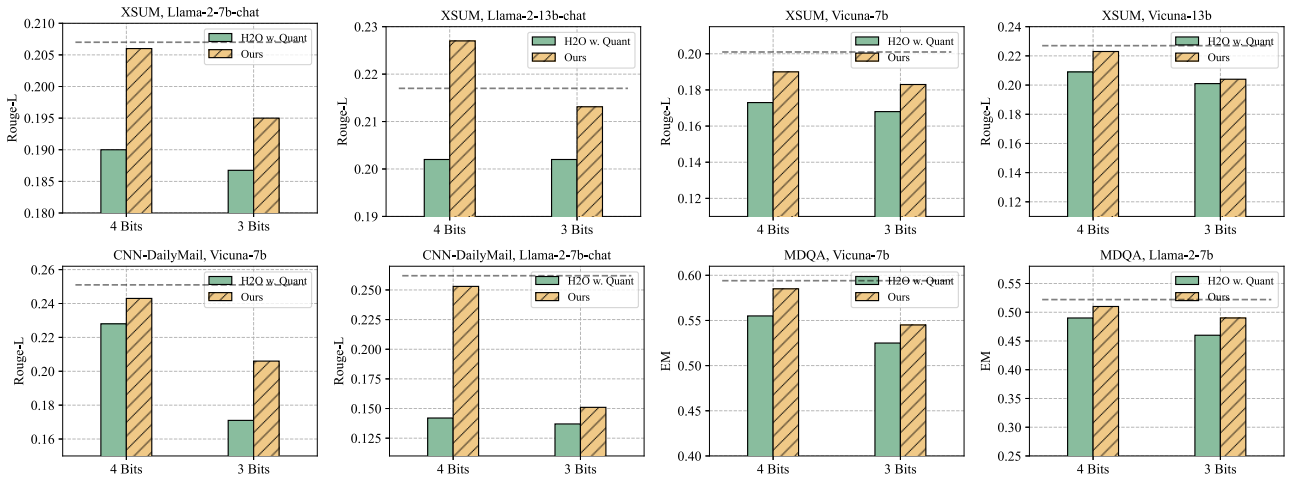

Figure 7. Results of different compression methods on summarization (XSUM, CNN-DailyMail) and Multi-Document QA (MDQA) tasks where Rouge-L and EM score are used to assess the generation quality. The dashed line represents the performance of the full model.

5.2 End-to-End Results

To begin, we demonstrate that (i) Q-Hitter consistently outperforms the conventional approach in which sparsity and quantization are applied sequentially across various tasks and models; (ii) it preserves the full model quality with only 5% to 12.5% memory requirements.

Superior performance over sequential sparsity and quantization. In Figure 7, we present a comprehensive comparison between Q-Hitter and the baseline approach that first employs $\mathsf{H_2O}$ for KV embedding eviction, followed by token-wise quantization for additional compression. Several observations can be drawn: (i) *Performance Advantage*: Q-Hitter consistently outperforms the baseline method across a diverse range of tasks and models. Under the quantization bits of both 4 and 3 bits, Q-Hitter achieves substantial improvements, with up to 78.17% increase (*e.g.*, on CNN-DailyMail with Llama-2-7b-chat model). (ii) *Long-Context Understanding*: In the MDQA task, where the critical document's position is varied, Q-Hitter demonstrates stable and superior performance, underscoring its proficiency in long-context understanding. (Note that in our experiments, we change the position of critical documents to appear at the beginning, middle, and end of all documents, respectively, and report the average performance.) This is a crucial capability for LLMs, especially in tasks that require synthesizing information from extensive contexts. (iii) *Surpassing Full Model Performance*: Interestingly, under certain compression settings, Q-Hitter not only matches but even exceeds the performance of the uncompressed full model, for example, on {XSUM, Llama-2-13b-chat}, Q-hitter shows 4.6% improvement of Rouge-L score against the full model. This extra bonus shows the potential of Q-Hitter to enhance the generation ability of LLMs. These observations collectively highlight the efficacy of Q-Hitter as a robust framework for compressing KV cache in Large Language Models, achieving significant memory savings while maintaining or even enhancing, model performance across various generative tasks and models.

No performance loss with only 5% to 12.5% memory. Furthermore, as illustrated in Figure 7, Q-Hitter is capable of preserving the performance of the full model while applying a 4-bit quantization, resulting in an overall compression ratio ranging from $8\times$ to $20\times$ across various tasks. Here, we compute the composite compression ratio as the product of sparsity and quantization contributions: $\mathrm{c} = \mathrm{c}_s \times \mathrm{c}_q$, where c_s and c_q represent the compression achieved through sparsity and quantization, respectively.

The requisite amount of memory to maintain performance varies depending on the task, given the inherent differences in information redundancy across tasks. For example, more memory-intensive tasks, such as MDQA, require a larger memory allocation. In our experiments, we have set c_s to $5\times$ for XSUM, $2.5\times$ for CNN-DailyMail, and $2\times$ for MDQA. The results demonstrate the efficacy of Q-Hitter in compressing the KV cache, establishing it as a reliable solution for achieving memory-efficient large language model (LLM) inference.

5.3 Q-Hitter Enhanced Throughput Improvements

Having established Q-Hitter's ability to retain model quality with a compression ratio of up to $20\times$ on the KV cache memory, we now shift our focus to quantifying its corresponding throughput improvements.

Table 2. Throughput (token/s) comparison of our Q-Hitter with different inference systems. For the sequence length, "512+32" represents a prompt length of 512 and generation length of 32. Experiments are conducted on a single T4 GPU.

Seq. length	512+32		512+512		512+1024	
Model size	6.7B	30B	6.7B	30B	6.7B	30B
Accelerate	20.4	0.6	15.5	0.6	5.6	0.6
DeepSpeed	10.5	7.6	10.1	7.9	11.3	7.3
FlexGen	20.2	8.1	16.8	8.5	16.9	7.1
H_2O	35.1	12.7	51.7	18.8	52.1	13.8
Ours	47.3	15.1	68.7	20.0	57.2	16.1

Implementation details. We integrate Q-Hitter into FlexGen (Sheng et al., 2023b) and compare it with other leading inference systems, including DeepSpeed (Aminabadi et al., 2022), Hugging Face Accelerate (HuggingFace, 2023), FlexGen (Sheng et al., 2023b), as well as H_2O (Zhang et al., 2023b). For our experiments, we configure Q-Hitter with a sparsity ratio of 20% and a quantization precision of 4 bits, ensuring full performance maintenance as verified in the previous section. The tensors of KV are stored in the quantized format during I/O. Once the KV are loaded, we dequantize these tensors back to FP16 for computation. The primary goal of this quantization approach is to reduce the I/O costs incurred during the iterative decoding steps of LLM inference. The dequantization does introduce additional computational overhead. However, given that the I/O costs during LLM inference significantly outweigh computational expenses, the overall throughput is still markedly improved due to the savings on I/O costs. Moreover, the efficiency of our method could be further enhanced through integration with optimized CUDA kernels, such as AWQ (Lin et al., 2023) or using the MX data format (Rouhani et al., 2023).

The evaluation metric of throughput is the number of generated tokens per second, where the cost time includes both prompting and generation stages. The evaluations are conducted on a single T4 GPU with 208GB CPU DRAM and 1.5TB SSD, with an end-to-end approach that includes the time required for prefilling, decoding and identifying important and quantization-friendly tokens. In scenarios where the combined memory requirements of the model and KV cache exceed the GPU's capacity, CPU offloading will be turned on. To provide a comprehensive benchmark, we assess the throughput of various methods across a range of prompting and generation lengths. Note that we follow exact same setting as (Zhang et al., 2023b) but using DeepSpeed with version 0.10.3.

Results and Analysis. Table 2 demonstrates the throughput comparison of different methods, in which a clear throughput improvements of our Q-Hitter can be drawn. Since the unnecessary KV embeddings are evicted and the remaining parts are quantized to low-precision, the memory requirements are largely reduced which enables us to use a larger batch size and more flexibility of either CPU or disk offloading. For OPT-6.7B, DeepSpeed has a higher memory cost and uses slower CPU offloading, resulting in its lower throughput, while all other methods can fit the model into a single GPU. For OPT-30B, all methods turn on CPU offloading and Accelerate store the KV cache on the GPU, thus restricts the batch size they can use, leading to a relatively low throughput, while other systems offload it to CPU. Results shows our Q-Hitter can achieve up to $33\times$, $7\times$, $4\times$ and $1.3\times$ throughput improvements compared with Hugginface Accelerate, DeepSpeed, FlexGen and H_2O, respectively. Note that, our method build on the top of FlexGen, and achieves a further $4\times$ throughput improvements via evicting unnecessary tokens as well as quantization. Part of the improvements against DeepSpeed and Accelerate are credits to FlexGen.

5.4 Further Investigation and Ablation Study

In this section, we aim to further explore the effectiveness of Q-Hitter framework. Specifically, we will address the following research questions: *Q1*: Can we stream LLMs to infinite input lengths with Q-Hitter? *Q2*: What is the trade-off between retaining significant tokens and opting for quantization-friendly tokens? *Q3*: Does the Q-Hitter framework consistently surpass the performance of baseline methods across different inference scenarios? *Q4*: Key or Value, which one is more challenging for quantization?

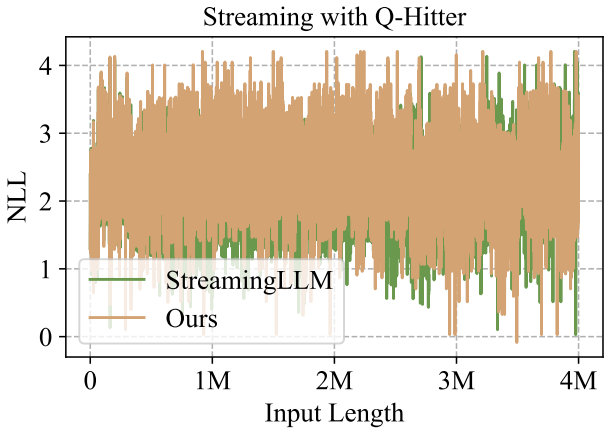

Figure 8. Streaming LLMs to infinite-input with Q-Hitter. The experiment is implemented on the test set of PG-19 (Rae et al., 2019) with Llama-2-7b. The baseline method StreamingLLM (Xiao et al., 2023b) employs the first four tokens and local tokens.

***A1*: Efficient Streaming of LLMs with Q-Hitter.** Beyond the challenges imposed by the extensive memory demands of the KV cache, another significant hurdle for Large

Language Models (LLMs) is their limited ability to tackle inputs with extremely long sequence lengths. Due to computational constraints, LLMs are typically pre-trained with a predetermined fixed sequence length, such as 4096 for Llama-2, resulting in their poor generalization to the inputs whose sequence lengths substantially exceed the pretrained length. Recent studies (Xiao et al., 2023b; Han et al., 2023) have made strides in enabling LLMs to handle sequences of infinite length. Inspired by that, we further explore the potential of our Q-Hitter framework to process infinite-length inputs. Following (Xiao et al., 2023b), We set the overall size of the KV cache to 2048 and allocate 128 for the tokens identified by our Q-Hitter algorithm, with the remaining 1920 space for local tokens. The input are fed iteratively via different chunks, avoiding the issue of large activation memory from the MLP blocks. As illustrated in the upper part of Figure 8, our Q-Hitter can effectively empower LLMs to manage sequences of up to 4 million tokens, showing the potential for infinite-length inputs. Note that StreamingLLM (Xiao et al., 2023b), H_2O (Zhang et al., 2023b), and our Q-Hitter are all capable of handling comparably lengthy ("infinite") sequences while the Q-Hitter algorithm is specifically designed for scenarios involving quantization and its primary strength lies in its ability to select tokens that are less impacted by the quantization.

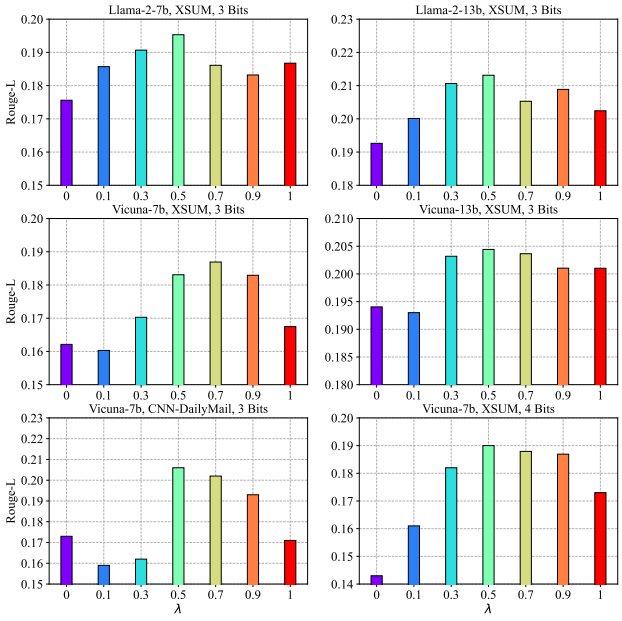

Figure 9. Performance of Q-Hitter with different value of balance factor λ.

***A2*: Ablation study of balance factor λ.** We conducted an ablation study to investigate the impact of varying balance factor (λ) values on the performance of our Q-Hitter framework. The results are presented in Figure 9. These experiments were performed with 3-bit quantization applied on the KV cache. As λ increases, we observe a trend where the generation quality initially improves, peaks, and then begins to deteriorate. This is because purely relying on accumulated attention scores will retain some hard quantized KV embeddings, leading to suboptimal performance. Conversely, an exclusive focus on quantization friendliness ($\lambda = 0$) can result in neglecting significant tokens, degrading the generation performance. Also, we can observe that a balance factor (λ) of 0.5 delivers the best performance, striking an optimal balance between attention significance and quantization friendliness. As such, we set λ to 0.5 as the default value for all the experiments.

***A3*: Consistent improvements under different shots of inference.** Table 3 showcases a comparative analysis between our Q-Hitter framework and the sequential approach of applying sparsity and quantization (H_2O w. 4 Bit). Each sample comprises k pairs of articles and summaries during the inference process to facilitate in-context learning. By varying the value of k, we assess the model's generative ability across different input sequence lengths. The results depict a consistent performance enhancement across various numbers of shots during inference. Taking the CNN-DailyMail task as an instance, our Q-Hitter framework is able to match the performance of the full model, achieving an improvement of up to 0.097 in terms of the Rouge-L score. Also, such improvement is more significant on zero-shot inference.

Table 3. Ablation study of the number of shots during inference. Experiments are conducted on Llama-2-7b-chat.

Settings	Methods	0	1	3	5
XSUM	Full	0.171	0.191	0.201	0.207
	H_2O w. 4 Bits	0.093	0.116	0.184	0.190
	Ours (4 Bits)	**0.152**	**0.171**	**0.194**	**0.202**
CNN-DailyMail	Full	0.234	0.243	0.254	0.258
	H_2O w. 4 Bits	0.132	0.202	0.163	0.228
	Ours (4 Bits)	**0.229**	**0.242**	**0.241**	**0.243**

***A4*: Quantizing the Key cache presents greater challenges compared to the Value cache.** Figure 10 compares the effects of quantizing either the key or value cache in isolation. As the number of quantization bits decreases, we observe a more rapid deterioration in performance when quantization is applied solely to the key cache, compared to the value cache. This observation underscores that the key cache is significantly more challenging to be quantized. A potential explanation for this phenomenon might be the broader value range of key embeddings(up to 70 for key cache in Pythia-70M while 12 for value cache), as shown in Figure 1. Larger value ranges are more prone to introducing

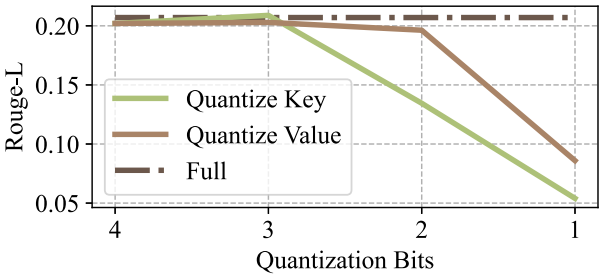

Figure 10. Comparison of quantization difficulty of key and value cache, where the horizontal axis represents the bits used for quantization. Results are collected from XSUM with Llama-2-7b-chat.

substantial quantization error, leading to a corresponding drop in performance.

6 Conclusion and Discussion

In this paper, we addressed the substantial memory and bandwidth challenges associated with deploying Large Language Models (LLMs) by providing a nuanced analysis of the interactions between sparsification and quantization in KV cache reduction. We first unveiled critical shortcomings in naive post-sparsification quantization approaches, particularly in low-bit scenarios: (i) KV cache quantization results in a significant accuracy degradation in the Heavy-Hitter selection process; (ii) the tokens identified by H_2O do not consistently exhibit compatibility with quantization. To address these challenges, we proposed Q-Hitter, a quantization-aware KV cache sparsification framework, that simultaneously takes "quantization friendliness" and accumulated attention scores into account. Q-Hitter successfully retains tokens that are crucial for maintaining LLM performance while being amenable to quantization, resulting in significant memory savings without compromising model efficacy. Furthermore, Q-Hitter can empower LLMs to effectively handle inputs with infinite sequence lengths. Our extensive experiments across various LLMs and tasks demonstrated the superiority of our Q-Hitter approach over previous post-sparsification quantization approaches, highlighting its potential to significantly enhance the affordability and efficiency of state-of-the-art LLM deployments.

7 Acknowledgement

This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344 and supported by the LLNL-LDRD Program under Project No. 24-ERD-010. S. Liu is funded by the Royal Society with the Newton International Fellowship.

References

Ainslie, J., Lei, T., de Jong, M., Ontañón, S., Brahma, S., Zemlyanskiy, Y., Uthus, D., Guo, M., Lee-Thorp, J., Tay, Y., et al. Colt5: Faster long-range transformers with conditional computation. *arXiv preprint arXiv:2303.09752*, 2023.

Aminabadi, R. Y., Rajbhandari, S., Zhang, M., Awan, A. A., Li, C., Li, D., Zheng, E., Rasley, J., Smith, S., Ruwase, O., et al. Deepspeed inference: Enabling efficient inference of transformer models at unprecedented scale. *arXiv preprint arXiv:2207.00032*, 2022.

Behdin, K., Acharya, A., Gupta, A., Keerthi, S., and Mazumder, R. Quantease: Optimization-based quantization for language models–an efficient and intuitive algorithm. *arXiv preprint arXiv:2309.01885*, 2023.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chee, J., Cai, Y., Kuleshov, V., and De Sa, C. Quip: 2-bit quantization of large language models with guarantees. *arXiv preprint arXiv:2307.13304*, 2023.

Cheng, W., Zhang, W., Shen, H., Cai, Y., He, X., and Lv, K. Optimize weight rounding via signed gradient descent for the quantization of llms. *arXiv preprint arXiv:2309.05516*, 2023.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin,

A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023a.

Dettmers, T., Svirschevski, R., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., and Alistarh, D. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023b.

Frantar, E. and Alistarh, D. SparseGPT: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Han, C., Wang, Q., Xiong, W., Chen, Y., Ji, H., and Wang, S. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*, 2023.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1135–1143, 2015.

HuggingFace. Hugging face accelerate. https://huggingface.co/docs/accelerate/index, 2023.

Jaiswal, A., Liu, S., Chen, T., and Wang, Z. The emergence of essential sparsity in large pre-trained models: The weights that matter. *arXiv preprint arXiv:2306.03805*, 2023.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Kim, S., Hooper, C., Gholami, A., Dong, Z., Li, X., Shen, S., Mahoney, M. W., and Keutzer, K. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 598–605, 1989.

Lee, C., Jin, J., Kim, T., Kim, H., and Park, E. Owq: Lessons learned from activation outliers for weight quantization in large language models. *arXiv preprint arXiv:2306.02272*, 2023.

Lee, D., Choi, J., Kim, J.-H., Noh, S. H., Min, S. L., Cho, Y., and Kim, C. S. Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE transactions on Computers*, 50(12): 1352–1361, 2001.

Li, L., Li, Q., Zhang, B., and Chu, X. Norm tweaking: High-performance low-bit quantization of large language models. *arXiv preprint arXiv:2309.02784*, 2023a.

Li, Q., Zhang, Y., Li, L., Yao, P., Zhang, B., Chu, X., Sun, Y., Du, L., and Xie, Y. Fptq: Fine-grained post-training quantization for large language models. *arXiv preprint arXiv:2308.15987*, 2023b.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023a.

Liu, P., Liu, Z., Gao, Z.-F., Gao, D., Zhao, W. X., Li, Y., Ding, B., and Wen, J.-R. Do emergent abilities exist in quantized large language models: An empirical study. *arXiv preprint arXiv:2307.08072*, 2023b.

Liu, Z., Oguz, B., Zhao, C., Chang, E., Stock, P., Mehdad, Y., Shi, Y., Krishnamoorthi, R., and Chandra, V. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023c.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*, 2023.

Mohtashami, A. and Jaggi, M. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

Mu, J., Li, X. L., and Goodman, N. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*, 2023.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

O'neil, E. J., O'neil, P. E., and Weikum, G. The lru-k page replacement algorithm for database disk buffering. *Acm Sigmod Record*, 22(2):297–306, 1993.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Rouhani, B. D., Zhao, R., More, A., Hall, M., Khodamoradi, A., Deng, S., Choudhary, D., Cornea, M., Dellinger, E., Denolf, K., et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

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

Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Fu, D. Y., Xie, Z., Chen, B., Barrett, C., Gonzalez, J. E., et al. High-throughput generative inference of large language models with a single gpu. *arXiv preprint arXiv:2303.06865*, 2023a.

Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Fu, D. Y., Xie, Z., Chen, B., Barrett, C., Gonzalez, J. E., et al. High-throughput generative inference of large language models with a single gpu. *arXiv preprint arXiv:2303.06865*, 2023b.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Wei, X., Zhang, Y., Li, Y., Zhang, X., Gong, R., Guo, J., and Liu, X. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*, 2023.

Wu, X., Yao, Z., and He, Y. Zeroquant-fp: A leap forward in llms post-training w4a8 quantization using floating-point formats. *arXiv preprint arXiv:2307.09782*, 2023.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023a.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023b.

Yuan, Z., Niu, L., Liu, J., Liu, W., Wang, X., Shang, Y., Sun, G., Wu, Q., Wu, J., and Wu, B. Rptq: Reorder-based post-training quantization for large language models. *arXiv preprint arXiv:2304.01089*, 2023.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V.,

et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhang, Y., Zhao, L., Cao, S., Wang, W., Cao, T., Yang, F., Yang, M., Zhang, S., and Xu, N. Integer or floating point? new outlooks for low-bit quantization on large language models. *arXiv preprint arXiv:2305.12356*, 2023a.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al. H _2 o: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv preprint arXiv:2306.14048*, 2023b.