CSE 256_FA24: KV Cache Compression via Token Reduction in Long-Context Scenarios

Neusha Javidnia
njavidnia@ucsd.edu

1 Introduction

Large Language Models (LLMs) have made remarkable progress in recent years, particularly in efficiently generating text. However, applications involving long-context scenarios, such as Retrieval-Augmented Generation (RAG), present unique challenges due to the high computational demands of managing extended input sequences. A promising solution to address these challenges is **KV cache compression**, which reduces the memory footprint and computational complexity while maintaining performance.

In this project, we explore a **token compression** approach based on two primary strategies. First, we utilize the state-of-the-art *LLMlingua* (Jiang et al., 2023) method to compress contexts at varying compression ratios. Second, we implement a summarization-based technique, where a dedicated summary model processes long contexts into concise representations before passing them to the target LLM. These methods aim to enhance the time efficiency and overall capability of LLMs in long-context scenarios by carefully balancing trade-offs between compression and performance.
Key Contributions

- Tested the `ChatGLM2-6B-32k` model on two long-context datasets comprising 200 and 150 samples. **DONE.**
- Compressed the long-context datasets using the *LLMlingua* tool with various target token settings and evaluated the performance of the compressed contexts on the `ChatGLM2-6B-32k` model. **DONE.**
- Implemented and tested a random token pruning strategy using *LLMlingua*'s tokenizer with different target token settings. **DONE.**
- Analyzed the trade-offs between different target token settings, focusing on timing and performance metrics. **DONE.**
- Attempted to compare the results with an alternative compression strategy using a summary model for context compression. **NOT DONE.** This comparison was not completed due to insufficient computational resources, as the summary model and the LLM could not fit on a single GPU.

2 Related work

2.1 KV cache compression

In transformers, the Key-Value (KV) cache stores the precomputed Key and Value matrices generated during the encoding process. This cache allows the model to reuse these matrices across multiple steps, significantly reducing computational overhead by avoiding repeated calculations. It is particularly useful for tasks requiring long-context processing or autoregressive generation, as it enables faster inference and efficient memory utilization by leveraging previously computed information.

KV cache compression has been explored across various dimensions of the Key and Value weights, including tokens, attention heads, hidden dimensions, and layers. For instance, compression along the attention heads has been demonstrated in methods like Multi-Query Attention (MQA) and Grouped Query Attention (GQA) (Ainslie et al., 2023). Multi-Query Attention (MQA) reduces memory and computation by sharing Key and Value projections across all heads. Grouped Query Attention (GQA) generalizes this by grouping multiple heads to share projections.

LLMLingua (Jiang et al., 2023) is a KV cache compression strategy designed to improve the efficiency of LLMs by reducing the token dimension. It employs a coarse-to-fine compression strategy that includes a budget controller to maintain semantic integrity under high compression ratios, a token-level iterative compression algorithm to model interdependencies between compressed contents, and an instruction tuning method for distribution alignment between language models.

LLMLingua utilizes a small language model to evaluate each token's perplexity within a prompt, identifying and removing tokens with lower perplexity values. This process effectively compresses the prompt by eliminating tokens that contribute less to the overall information entropy, thereby maintaining the essential context while reducing prompt length. This enables up to 20x compression, thereby accelerating model inference and reducing computational costs.

The Token Compression Retrieval Augmented LLM (TCRA-LLM) (Liu et al., 2023) introduces another method for compressing the KV cache along the token dimension through two techniques. The first employs a fine-tuned T5-based model to summarize text, condensing information into a shorter, more concise form for input to the target LLM. The second utilizes semantic compression by selectively removing low-impact words, effectively reducing input size while preserving critical content.

Context-aware Prompt Compression (CPC) (Liskavets et al., 2024) is a sentence-level compression technique that enhances LLMs' efficiency by removing less relevant sentences from prompts. It employs a context-aware sentence encoder trained to assign relevance scores to each sentence concerning a specific question. This encoder is trained using a dataset of questions paired with positive sentences (relevant to the question) and negative sentences (irrelevant). By filtering out sentences with lower relevance scores, CPC effectively reduces prompt length while preserving essential information, leading to faster inference times and improved performance compared to token-level compression methods.

H_2O (Heavy-Hitter Oracle) (Zhang et al., 2023) identifies and retains "heavy-hitter" tokens—those significantly influencing attention scores—within the KV cache while discarding less impactful tokens. This selective retention reduces memory usage and computational load without compromising performance. *H_2O* achieves up to a 29-fold throughput improvement over existing inference systems when retaining 20% of tokens.

3 Dataset

In this project, we selected two datasets from the LongBench benchmark (Bai et al., 2024)—Qasper and MultiFieldQA-en—comprising 200 and 150 samples, respectively. Both datasets are designed for single-document question answering (QA) tasks, where precise answers to questions are derived from information contained within a single document. This task requires a system to comprehend the document's content and accurately extract or infer relevant information to respond effectively.

LongBench is a comprehensive benchmark that evaluates the capabilities of large language models (LLMs) in understanding long contexts across multiple tasks and languages. Qasper consists of questions paired with lengthy contexts, averaging 3,599 words per sample and reaching up to 14,640 words. MultiFieldQA-en features contexts with an average length of 4,538 words and a maximum of 10,330 words. These substantial context lengths pose significant challenges for LLMs, necessitating robust strategies for processing and comprehending extensive textual information.

As shown in Table 1, I present two examples from the single-document question answering (QA) task, which involves answering a question based on a given context. According to the original LongBench paper, Qasper and MultiFieldQA-en datasets are evaluated using the F1 score. F1 score measures the balance between precision (the fraction of relevant instances among retrieved results) and recall (the fraction of relevant instances retrieved from the total relevant results). It is calculated as the harmonic mean of precision and recall, providing a single metric to evaluate the model's accuracy in tasks like question answering or classification. The F1 scores of various models are shown in Table 2, highlighting the performance differences across architectures. In this project, we use the `ChatGLM2-6B-32k` model, which achieves a relatively modest F1 score. This is attributed to the inherent complexity of long-context scenarios, where maintaining relevant information over extended sequences poses significant challenges for large language models. However, such tasks offer valuable insights into the model's ability to handle intricate long-context reasoning and comprehension.

Table 1: Overview of Selected Datasets (Qasper and MultiFieldQA-en) from Hugging Face

Dataset: Qasper **Input:** What neural network modules are included in NeuronBlocks? **Context:** Introduction Deep Neural Networks (DNN) have been widely employed in industry for .... **Length:** 1676 **Answers:** Embedding Layer, Neural Network Layers, Loss Function, Metrics
Dataset: MultiFieldQA-en **Input:** What is the effect of the proximity of superconductivity on the Kondo effect? **Context :** Introduction The exchange interactions control the magnetic order and properties of a vast.... **Length:** 5009 **Answers:** It tends to suppress the Kondo effect.

Table 2: F1 Scores for Qasper and MultiFieldQA-en across Models from LongBench (Bai et al., 2024)

Model	Qasper	MultiFieldQA-en
GPT-3.5-Turbo-16k	43.3	52.3
LLaMA2-7B-chat-4k	19.2	36.8
LongChat-v1.5-7B-32k	27.7	41.4
XGen-7B-8k	18.1	37.7
InternLM-7B-8k	16.7	23.4
ChatGLM2-6B	22.5	35.0
ChatGLM2-6B-32k	31.5	46.2
Vicuna-v1.5-7B-16k	26.1	38.5

4 Baselines

In this project, two baselines were established. The first baseline involves using the original, uncompressed context combined with the corresponding question as input to the target LLM, `ChatGLM2-6B-32k`. The second baseline employs a random token pruning strategy, where tokens are randomly removed from the context to reach a specified number of tokens before inputting it into the target model. For the random pruning strategy, *LLMLingua*'s tokenizer was utilized to ensure consistency in tokenization criteria and maintain comparability with the main compression method, *LLMLingua*, particularly in terms of the compression rate. This setup allows for a direct evaluation of the performance (F1 score) and latency differences between structured compression and random pruning.

5 My approach

In this project, I have utilized the *LLMLingua* tool to compress input contexts into various target token lengths for evaluation with the target LLM, `ChatGLM2-6B-32k`, which has not been assessed in the original *LLMLingua* paper. This compression technique significantly reduces the size of the input context while retaining its most relevant information, ensuring that the essential meaning and critical details are preserved. Once compressed, the compact representation is paired with the corresponding input question and provided as input to the target large language model. This approach is anticipated to outperform random token pruning by preserving semantic integrity while also offering faster inference compared to processing the full context. However, it may slightly underperform compared to using the uncompressed, full context. Importantly, *LLMLingua* operates in a training-free manner, requiring no additional training or fine-tuning. Similarly, the target LLM, `ChatGLM2-6B-32k`, is utilized directly for inference without any further fine-tuning or modification.

Additionally, I explored the possibility of an alternative compression strategy using a summary model for context compression. However, this comparison could not be completed due to insufficient computational resources, as the summary model `long-t5-tglobal-large` and the target LLM `ChatGLM2-6B-32k` exceeded the memory capacity of a single GPU. This approach is inspired by the method described in (Liu et al., 2023), which uses a fine-tuned summary model. I hypothesize that this method could outperform both the original context baseline and random token pruning, offering a balance between context relevance and computational efficiency.

The implementation of the baselines and methods in this project is distributed across several files:

- `full_ds.py`: Associated with the baseline that uses the complete, uncompressed context as input to the target LLM.
- `llm_lingua_ds.py`: Implements the approach using the *LLMLingua* tool to compress the context to various target token lengths.
- `random_prune.py`: Handles the baseline

where tokens are randomly pruned to achieve a specified token length for the input context.

- `summary_ds.py`: Intended for implementing a summarization-based compression strategy, though this method was not fully explored due to computational constraints.

Each file corresponds to a distinct methodology for handling long-context scenarios, enabling a systematic comparison of their performance in terms of accuracy and inference efficiency.

The experiments were conducted on a high-performance computing system equipped with four NVIDIA RTX A6000 GPUs, each providing 48 GB of memory, running with NVIDIA driver version 535.183.01 and CUDA version 12.2.

LLM models with FP16 precision are used to improve efficiency by reducing memory usage and accelerating GPU computations, enabling larger models and batch sizes. FP16 supports mixed-precision training, combining its speed with FP32 stability to optimize performance and resource utilization. This approach is ideal for my project, enhancing inference efficiency while maintaining accuracy, particularly for handling large-scale contexts.

6 Results

In this section, we analyze the results in terms of F1 score and end-to-end latency for all the discussed methods. Table 4 presents the F1 scores and average end-to-end latency of `ChatGLM2-6B-32k` on the MultifieldQA_en and Qasper datasets. As shown, the performance on the Qasper dataset is significantly lower compared to MultifieldQA_en. Due to these unsatisfactory results, the Qasper dataset is excluded from the main comparisons.

Figure 2 illustrates the average end-to-end latency comparisons across different approaches. As expected, both *LLMLingua* and random token pruning achieve faster inference latencies compared to using the full context, demonstrating the efficiency of these compression methods. For instance, with a target token count of 2000, the average end-to-end latency for *LLMLingua* is 0.68 seconds, compared to 0.89 seconds for random token pruning and 2.63 seconds for the full context. This represents a speedup of approximately 3.85x for *LLMLingua* and 2.94x for random token pruning over the full context. Table 3 further highlights

Table 3: Speed-up Comparisons for Random Pruning and LLMLingua

Target Tokens	Random Pruning Speed-up	LLMLingua Speed-up
500	3.38x	5.05x
1000	3.69x	4.55x
2000	2.94x	3.85x
3000	2.50x	3.36x
5000	2.24x	2.48x
7000	2.24x	1.96x
10000	1.42x	1.63x

how more compression leads to greater speedups.

The latencies for *LLMLingua* are consistently lower than those for random token pruning, underscoring its effectiveness in compressing context while preserving relevant information. This advantage is attributed to *LLMLingua*'s structured compression strategy, which prioritizes semantically meaningful tokens, resulting in reduced computational overhead and more efficient processing. These results emphasize the importance of intelligent compression techniques like *LLMLingua* for handling long-context scenarios.

Figure 1 shows the F1 scores of *LLMLingua*, random token pruning, and full context usage across various target token lengths. As expected, there is a performance drop in both *LLMLingua* and random pruning approaches compared to using the full context due to the loss of some key information during compression. At lower target token lengths, *LLMLingua* consistently achieves higher F1 scores than random token pruning, demonstrating its ability to retain more semantically important information.

The F1 score performance decreases significantly at lower target token lengths due to the challenge of preserving essential context in highly compressed inputs. At higher target token counts, the performance gap between *LLMLingua* and random pruning narrows as both methods retain more tokens, which include most of the critical information. Notably, the average token length of the full context in MultifieldQA_en dataset is 8057.62, with a minimum of 1372 and a maximum of 17,727 tokens.

In summary, the results showcase the effectiveness of compression methods like *LLMLingua* in improving inference speed and latency while achieving a balance between performance and efficiency. Although compression comes at the cost of reduced F1 scores, these methods significantly enhance the feasibility of long-context processing.

Table 4: Performance and average end-to-end latency of `ChatGLM2-6B-32k` on two datasets.

Dataset	**Samples**	**F1 Score**	**Avg. Latency (s)**	**Latency Range (s)**
MultifieldQA_en	150	29.77	2.63	0.33–12.70
Qasper	200	12.35	1.82	0.42–12.69

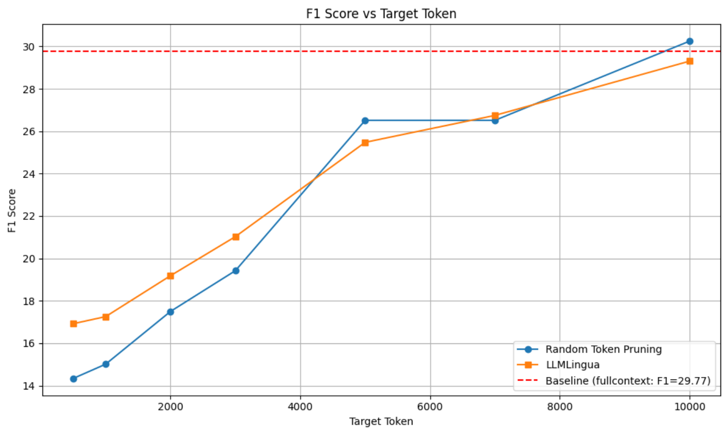

Figure 1: F1 score vs. target token count using *LLMLingua* and Random Token Pruning approaches compared to Full Context on MultifieldQA_en dataset.

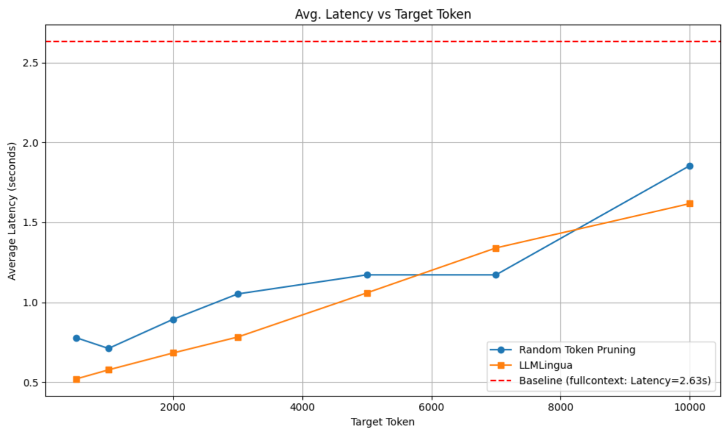

Figure 2: Comparison of average end-to-end latency across Full Context, *LLMLingua*, and Random Token Pruning methods on MultifieldQA_en dataset.

Table 5: Performance and average end-to-end latency of *LLMLingua* and Summary Model on the Qasper dataset. Since the F1 scores are unsatisfactory, this table is not considered for the main analysis.

Run	**F1 Score**	**Avg. Latency (s)**	**Avg. Compression (%)**
LLMLingua	9.48	0.6963	66.30
LLMLingua	9.51	0.9592	52.81
LLMLingua	9.31	0.7800	36.01
Summary Model	6.53	1.6126	88.36

7 Error analysis

Table 5 provides additional runs on the Qasper dataset using *LLMLingua* and the summary-based model. However, since the F1 scores on this dataset with `ChatGLM2-6B-32k` remain unsatisfactory, these results are not considered for the primary analysis. Generally, the `ChatGLM2-6B-32k` model performs poorly on Qasper primarily due to its difficulty handling highly specific, context-dependent questions that require deep understanding and synthesis of scientific content. The failed examples often involve multi-hop reasoning across multiple sections of a document, interpreting domain-specific terminology, or connecting implicit ideas not explicitly stated in the text.

Furthermore, at lower target token counts, both random pruning and *LLMLingua* exhibit poor performance. These failures are primarily attributed to aggressive compression, which compromises critical semantic and syntactic information necessary for accurate reasoning and understanding.

8 Conclusion

In conclusion, compressing long contexts significantly improves processing speed but comes at the cost of reduced performance. The extent of compression must be carefully evaluated based on the specific task requirements to balance reasonable performance with time efficiency. Identifying an optimal token compression approach that minimizes performance loss remains a challenge. As a future direction, I aim to explore incorporating additional dimensions of Key-Value compression, such as Multi-Query Attention (MQA), into this method to further enhance speed and memory efficiency. However, this would require access to robust computational resources and a sufficient number of test cases to ensure comprehensive evaluation.

9 Acknowledgements

- ChatGPT was used to assist in formatting my results into LaTeX tables.
- ChatGPT helped me in selecting an appropriate summary model for long-context tasks.
- ChatGPT was used to correct grammatical errors in my written text.

References

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. (2023). Gqa: Training generalized multi-query transformer models from multi-head checkpoints.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. (2024). Longbench: A bilingual, multitask benchmark for long context understanding.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. (2023). Llmlingua: Compressing prompts for accelerated inference of large language models.

Liskavets, B., Ushakov, M., Roy, S., Klibanov, M., Etemad, A., and Luke, S. (2024). Prompt compression with context-aware sentence encoding for fast and improved llm inference.

Liu, J., Li, L., Xiang, T., Wang, B., and Qian, Y. (2023). Tcra-llm: Token compression retrieval augmented large language model for inference cost reduction.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z., and Chen, B. (2023). H_2o: Heavy-hitter oracle for efficient generative inference of large language models.