Streaming LLM with Dynamic Global Context

Aaron Li, Yilin Wang, Mingyuan Ma

This is the final project for the course EfficientML @ MIT, Fall 2023. All source code can be accessed at `https://github.com/aaron-jx-li/augmented_streamingLLM`.

1 Introduction

The development of large language models (LLM) has greatly promoted rapid progress in the field of natural language processing (NLP) and beyond, especially those models based on the Transformer architecture [Vaswani et al., 2017]. However, as we push the boundaries of what these models can achieve, we encounter a series of challenges that will be critical to the next leap forward in Deep Learning research and applications.

The first major challenge is the heavy memory usage during decoding. Transformer-based LLM caches Key and Value state to make the generation process more efficient without the needs to recalculate all previous K, V scores. This mechanism, however, results in large memory requirements. Furthermore, the second challenge lies in the inherent design of LLMs (using Attention mechanism), which generally limits them to processing text sequences that are no longer than the length of the training sequence.

To address these challenges, researchers at MIT Han Lab introduce streaming LLM [Xiao et al., 2023], a novel framework for deploying LLM in streaming environments. This approach solves the problem of high memory consumption and extends the model's ability to handle sequences of variable length. The key of streaming LLM is its ability to operate with a limited attention window, allowing efficient processing of infinitely long sequences without the need for additional fine-tuning.

However, streaming LLM also has its limitations. Its design focuses primarily on initial tokens (named as attention sinks in the paper) and tokens within a fixed-size recent window, essentially imposing a performance upper bound. Once tokens are evicted from this window, they cannot be retrieved, even if they have significant semantic importance and the content may be needed in the future. This design choice, while optimizing memory usage, also runs the risk of losing contextually relevant information, which can compromise the quality of the generated content.

There are several ways to solve this issue, including retrieval-augmented generation (RAG) [Lewis et al., 2021]. RAG enriches the model's context by allowing it to retrieve relevant information from large corpora, thus improving the quality and relevance of its output. However, directly applying RAG to streaming LLM poses significant computational challenges, because it needs to search over relevant context, which might be compute-intensive. Also the intermediate context might require lots of memory.

To mitigate those issues listed above, we propose a new approach called Dynamic Global Context (DGC) to enhance the streaming LLM framework. DGC is inspired by Unlimiformer [Bertsch et al., 2023], which introduces an innovative approach to enhance the retrieval capabilities of LLM. The Unlimiformer model utilizes k-nearest neighbor (kNN) indexing to enable retrieval from the entire input sequence, rather than relying on truncation. This design guides us to propose the enhanced streaming LLM framework, where we aim to maintain the top k candidates in the cache during inference. Our approach carefully balances the need for context preservation with the computational efficiency required to process long sequences in a streaming environment. To be specific, our approach focuses on selectively retaining important information in the KV cache, thereby addressing the challenge of context forgetting in long sequence generation. The DGC algorithm is designed to evaluate the contextual importance of tokens, ensuring that important information is preserved

beyond the immediate visibility window. This mechanism enables the model to maintain continuity and coherence when processing large sequences of text, a key aspect that is often overlooked in streaming LLM.

We further delve into the architectural details of the DGC algorithm, highlighting its innovative memory management strategies, including a double-cache system and a smart eviction protocol. This approach allows global context to be handled dynamically, ensuring the model remains agile and efficient. Our empirical evaluation on a SOTA Transformer-based model demonstrates the efficient of the DGC algorithm in preserving relevant context without compromising the inherent advantages of Streaming LLM.

In summary, our project has following contributions:

- **Dynamic Global Context for Context Retention** DGC innovatively addresses the challenge of context forgetting in long sequence generation, which is the issue that existed in Streaming LLM. In other words, we balance the efficiency and performance for text generation.
- **Dual-Cache Management** Our research introduces a unique dual-cache system for memory management in streaming LLMs. This system not only retains the advantages of the existing streaming LLM framework but also incorporates a dynamic cache for global context, which together with DGC, balancing the need for context retention with the constraints of computational efficiency.

2 Related Works

2.1 Streaming LLM

Modern Large Language Models (LLMs), mostly Transformer-based model Vaswani et al. [2017], are facing two major challenges: extensive memory usage during decoding due to caching Key and Value states, and the inability of LLMs to handle text longer than their training sequence length. To solve those two issues, experts from Han Lab at MIT presents a novel approach for deploying LLMs in streaming environments. It addresses key challenges such as high memory consumption during decoding and the limited ability of LLMs to process long texts. The research introduces streaming LLM Xiao et al. [2023], a framework that enables LLMs trained with finite attention windows to handle infinite sequence lengths efficiently. This is achieved without requiring additional fine-tuning. More importantly, the paper also shows that the phenomenon of attention sink arises from the model's strong focus on the initial tokens as an "attention sink," regardless of their semantic significance. Attention sinks are important as they significantly influence the distribution of attention weights in language models. This focus on early tokens helps stabilize the model's performance, especially in processing longer text sequences. By acting as a focal point for the attention mechanism, these sinks play a critical role in managing the model's resources and ensuring efficient performance in streaming contexts.

2.2 Retrieval Augmented Generation (RAG)

Despite its faster inference speed and low memory consumption, one intrinsic shortcoming of streaming LLM is that it only attends to the initial attention sink and a fixed-size recent window, while evicting all tokens in the middle. This design by nature puts an upper bound on its generation performance: once evicted, those tokens cannot be retrieved in the future, even if they are semantically important or have relatively high attention weights.

Retrieval Augmented Generation (RAG) is one potential approach to address this issue. RAG is first introduced by Lewis et al. [2021] to combine pretrained parametric models and non-parametric memory for language generation. For example, it's possible to augment the memory of a pretrained seq2seq model with a dense vector index of Wikipedia, accessed with a pre-trained neural retriever. Another recent work that's more closely connected to streaming LLM's context is Unlimiformer [Bertsch et al., 2023], which offloads the cross-attention computation to a single k-nearest-neighbor (kNN) index that could be utilized by each head in each decoder layer. Such design allows retrieval from the entire input sequence instead of truncating.

Our approach is inspired by Unlimiformer in that we could maintain the top-k candidates (key-value pairs) in the cache during inference, but simply searching for closest keys for each token to

be generated would be computational prohibitive and thus undermine streaming LLM's ability to efficiently process infinitely long sequences.

3 Methods

3.1 Dynamic Global Context

As mentioned before, it's possible for streaming LLM to evict important previous tokens, and an intuitive solution to mitigate this issue is to have a selection mechanism that's able to detect important information and keep the corresponding tokens in our KV cache, even they are beyond the recent window size. To this end, we propose to augment the original KV cache with an additional component named Dynamic Global Context (DGC).

Similar to the recent context and attention sink in the original Streaming LLM framework, DGC also has a predefined context size, which enables the model to store the important information that would otherwise be evicted in Streaming LLM. Compared with conventional dense attention, it improves the inference efficiency as it only attends to a fixed number of tokens instead of all previous tokens (which makes the runtime during inference of generating each new token to be $O(T)$, where T is length of previous tokens). Specifically, DGC employs an evaluation mechanism to score tokens based on their contextual importance, enabling the preservation of vital information even beyond immediate visibility. This ensures continuity in learning as the model traverses through vast sequences of data.

As part of its operation, DGC maintains a global context cache — a strategic reservoir of tokens with significant long-term relevance. Tokens within this cache are scored and ranked such that the most pertinent information are maintained to inform future generations. Given the hidden representations of Attention Sinks (H_{sink}), Global Context (H_{global}), and Local Context (H_{local}), we first concatenate these representations and then apply the learned weight matrices to obtain the Query (Q), Key (K), and Value (V) matrices:

$$H = \text{Concatenate}(H_{\text{sink}}, H_{\text{global}}, H_{\text{local}}) \tag{1}$$

$$Q = W_Q H, \quad K = W_K H, \quad V = W_V H \tag{2}$$

The attention weights (A) are then calculated using the scaled dot-product conventional attention mechanism Vaswani et al. [2017]:

$$A = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{3}$$

where d_k is the dimension of the Key vectors. This formulation ensures that the attention mechanism takes into account the composite information from the Attention Sinks, Global Context, and Local Context, thereby capturing a richer representation of the sequence history. Since now our attention score does not depends on the length of previous tokens as we restricted it to a fixed number of tokens, the run time to generate each new token would become $O(1)$ respective to the length of previous tokens and the memory consumption is also a fixed amount, and thus successfully inherits the advantages of Streaming LLM but also mitigate the forgetting issue after a long sequence of generation.

A predefined hyperparameter dictates the size of the dynamic cache to store previous important tokens to minimize the loss of global context, balancing the breadth of context retention with computational feasibility. When the cache reaches its capacity, it employs an eviction strategy which replaces the least important tokens with new and potentially more informative ones from the recent context.

The detailed algorithm for dynamically managing the global context is included in Algorithm 1.

This process exemplifies the essence of continuous learning and adaptation. DGC is designed to learn which tokens are worth remembering and which can be forgotten, mimicking a form of selection that human readers naturally possess. Such an approach allows the model to remain agile and efficient, circumventing the pitfalls of memory saturation that plague conventional models.

Algorithm 1 Dynamic Global Context

Require: Tokens stream, GC, GC_score = [], []

Hyperparameter: GC_size
while processing tokens **do**
    Move earliest_token_in_recent_tokens to GC
    Update GC_score
    if len$(GC) >$ GC_size **then**
        Evict least important token according to GC_score
    end if
end while

We note that Algorithm 1 is inherently generalizable. It can be adapted to different model sizes and types, and our experiments are conducted on the latest Transformer-based model, making it to be easily adopted by other LLM applications.

3.2 Importance Metric

As briefly mentioned earlier, using unsupervised learning methods to predict the importance of previously seen key vectors for each incoming token during inference is not computationally efficient. In this project, we propose to use Term Frequency - Inverse Document Frequency (TF-IDF) [Sparck Jones, 1988] as an external importance metric to maintain our dynamic global context.

TF-IDF is a widely used statistical method for NLP tasks that measures how important a term is within a sentence relative to an entire text corpus. Specifically, the importance of a particular word in the current sentence is given by:

$$\text{TF-IDF(w)} = \frac{\text{Number of w in current sentence}}{\text{Number of w in the entire corpus}} \cdot \log\left(\frac{\text{Total number of sentences}}{\text{Number of sentences contain w}}\right) \quad (4)$$

Intuitively, a word would be considered important when it occurs a lot in a given document/sentence and rarely in others. As TF-IDF scores are easy to compute and it's trivial to map the word importance scores to token importance scores, adding the extra DGC component in the KV cache doesn't slow down the inference substantially.

4 Experimental Setup

4.1 Data

WikiText [Merity et al., 2016] This dataset contains texts scrapped from Wikipedia. Each sample are consecutive section on a Wikipedia page. This suggests that each sample is relatively independent, though there may be some correlations. We respectively use the first 100, 1000, and all samples in this dataset to evaluate the perplexity of the language models.

BookCorpus [Zhu et al., 2015] This dataset contains texts from 11038 books. In this dataset, samples are consecutive sentences in the book, which means there exists a stronger correlation between samples. We respectively use the first 1000 and 5000 samples in this dataset to evaluate the perplexity of the language models.

4.2 Evaluation

We evaluate the predictive performance (in terms of perplexity) of a pretrained Llama2-7B-chat model equipped with our augmented KV cache.

To test the effectiveness of the dynamic global context, we fix the total size of the KV cache to be 511. In all experiments, we use an attention sink of size 1, and split the remaining 510 slots into dynamic global context and recent context of different sizes, as we also want to analyze the effect of this tradeoff between different component sizes within the cache.

During inference, we update the KV cache such that in each iteration the least recent token in the recent context is moved to the dynamic global context, which always keeps the top-k (where k is the size of the DGC) most important tokens in history based on TF-IDF scores.

5 Results

Table 2 shows the results of our model on WikiText. From the table, we see that our model does outperform streaming LLM (first row) in some specifications. Overall, we see that the perplexity decreases as the number of samples used in the evaluation increases. We also note that we only need a small amount of global context, as we observe performance degradation when we trade more recent context for global context when the number of global contexts is large.

GC Size	RC Size	Sink?	Perplexity(1000)	Perplexity(All)
0	510	✓	8.44	7.93
60	450	✓	**8.37**	7.89
110	400	✓	8.48	**7.87**
160	350	✓	8.61	8.03
210	300	✓	8.78	8.05
255	255	✓	8.96	8.24

Table 1: Perplexity on BookCorpus. perplexity(n) means evaluating using the first n samples in the data. GC and RC refer to the global context and the recent context. Sink refers to whether we include the attention sink (1 token).

Table 2 shows the results of our model on BookCorpus. We overall see a higher perplexity on BookCorpus than WikiText. One hypothesis is that texts from the BookCorpus are more sophisticated and less formal, making it harder for the LLM to predict the next token. Our model achieves larger performance gain on BookCorpus than on WikiText, as our framework outperforms streaming LLM (first row) in *all* specifications and dataset sizes. One potential explanation is that the texts in Bookcorpus are consecutive sentences in books, which makes them more inter-connected and renders the global context (which contains distant past information) necessary for next-token-prediction.

We see that replacing part of the recent context with global contexts yields better performance, which indicates that the global contexts are more important than the most distant part of the recent context. Nevertheless, in the optimal combinations, the size of the global context is still less than the recent contexts. Beyond the optimal point, increasing the global context size leads to performance degradation.

GC Size	RC Size	Sink?	Perplexity(1000)	Perplexity(5000)
0	510	✓	20.13	19.50
60	450	✓	19.82	**19.28**
110	400	✓	19.61	19.31
160	350	✓	**19.51**	19.32
210	300	✓	19.62	19.32
255	255	✓	19.97	19.54

Table 2: Perplexity on BookCorpus. perplexity(n) means evaluating using the first n samples in the data. GC and RC refer to the global context and the recent context. Sink refers to whether we include the attention sink (1 token).

6 Future Works

6.1 Attention-Based Token Importance

In this study, we utilize TF-IDF as the token importance scoring function. However, this metric is not contextualized.

We also propose an alternative algorithm based on the attention-oriented token importance metrics. When attending to the next token, the model record the attention weights for RC and GC, and we update the score for RC and GC in `all_Score` by taking the average attention score over *all* tokens that each token in RC&GC has attended to. That is what line 5 intends to achieve. Then, we pick the global context (GC) to the top k tokens with the highest score. There are several issues with this algorithm. First, we need to store the K, V values of *all* tokens in memory, which is expensive. We might try to alleviate this issue by offloading them to CPU memory and completing the GC selection process in the CPU since it cannot be parallelized. Further, the `topk` function at the end could be very expensive depending on the data structure. Some issues need to be addressed for this algorithm to be deployable, and we leave this to future work.

Algorithm 2 Attention-Based Token Importance

Require: GC, all_Score, all_tokens_kv = [], []
1: Hyperparameter: limit, recent_size
2: **while** processing tokens **do**
3: Add next_token to GC
4: attn weights = Attention(next_token and recent tokens & global context)
5: all_score[RC & GC] <- (new_attn weights + all_score[RC & GC] * n) / (n+1)
6: Add next_token k, v to all_tokens_kv
7: GC <- topk(all_tokens_kv, score = all_score)
8: **end while**

6.2 Token-Cluster representation via Incremental Learning

The algorithm presented in the pseudocode below 3 4 incorporates the concept of incremental learning Kirkpatrick et al. [2017] to optimize the retention of important contextual information without overwhelming the model's memory capacity. This approach is a direct response to an identified problem in streaming LLM, where intermediate tokens are evicted, potentially resulting in valuable context being lost. Inspired by the principles presented in an algorithm named iCaRL for incremental learning Rebuffi et al. [2017], our algorithm strategically manages token embeddings through a dynamic clustering mechanism.

The key to the algorithm is its initial phase, which determines the number of clusters used to divide the nearest token by dividing its length by a factor (for example, 5 in our case). Then pass to a function Φ, a feature extractor, identifies salient features of the data and encodes them into vectorized representations if we treated model (before the last fully connected layer) as a feature extractor. These vectors are then clustered using the k-means++ method Arthur and Vassilvitskii [2007], where the centroid of each cluster is the mean embedding, representing the global context. Each cluster are represented by a fix number (for example, 255 in our case) of relevant token features (the output of Φ). As the token flow continues, the algorithm permanently adapts to the global context by recalibrating the cluster using incoming token embeddings.

Algorithm 3 Prediction Function

function PREDICTION
n_clusters $\leftarrow$ len(recent_tokens)$//5$
clusters $\leftarrow \{\}$
clusters.put(k_means(Φ(recent_tokens), $n\_clusters$))
while processing **do**
clusters_embedding $\leftarrow$ []
for cluster in clusters.keys() **do**
clusters_embedding.append(clusters[cluster].mean())
end for
next_token_embed $\leftarrow \Phi$(attention_sinks, $clusters\_embedding$, recent_tokens)
update_cluster(next_token_embed, clusters)
next_token $\leftarrow$ logit($next\_token\_embed$)
end while
end function

Algorithm 4 Update Cluster Function

```
function UPDATE_CLUSTER(next_token_embed, clusters)
    cluster_scores ← []
    for cluster in clusters.keys() do
        token_cluster_similarity ← sim(clusters[cluster].mean(), next_token_embed)
        cluster_scores.append(token_cluster_similarity)
    end for
    clusters[argmax(cluster_scores)].push(next_token_embed)
    if clusters[argmax(cluster_scores)].size() > 255 then
        clusters[argmax(cluster_scores)].pop()
    end if
end function
```

7 Conclusion

Our algorithm, Dynamic Global Context (DGC), presents an innovative method for optimizing Large Language Models for streaming applications. The core of this method lies in its unique approach to handling the inherent limitations of Transformer-based LLMs, such as memory inefficiency and incapacity to process texts longer than the model's training length. Adding on streaming LLM, our algorithm offers a solution that enables efficient infinite sequence processing without additional training requirements. The DGC extends the capabilities of Streaming LLM by managing a global context cache that dynamically preserves essential tokens based on DGC's scoring system, ensuring continuity and coherence in understanding extensive text sequences. The DGC algorithm's memory management strategy is not only practical but also effective, balancing context retention with computational resources. This balance is achieved through an smart eviction strategy that maintains token's importance and mode's efficiency. From experiments and results, we provide compelling evidence of the effectiveness of the DGC algorithm, making a significant contribution to the field of NLP and beyond.

References

David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. Unlimiformer: Long-range transformers with unlimited length input, 2023.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

Karen Sparck Jones. *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, page 132–142. Taylor Graham Publishing, GBR, 1988. ISBN 0947568212.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.