ONE PASS STREAMING ALGORITHM FOR SUPER LONG TOKEN ATTENTION APPROXIMATION IN SUBLINEAR SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention computation takes both the time complexity of $O(n^2)$ and the space complexity of $O(n^2)$ simultaneously, which makes deploying Large Language Models (LLMs) in streaming applications that involve long contexts requiring substantial computational resources. In recent OpenAI DevDay (Nov 6, 2023), OpenAI released a new model that is able to support a 128K-long document, in our paper, we focus on the memory-efficient issue when context length n is much greater than 128K ($n \gg 2^d$). Considering a single-layer self-attention with Query, Key, and Value matrices $Q, K, V \in \mathbb{R}^{n \times d}$, the polynomial method approximates the attention output $T \in \mathbb{R}^{n \times d}$. It accomplishes this by constructing $U_1, U_2 \in \mathbb{R}^{n \times t}$ to expedite attention $\mathsf{Attn}(Q, K, V)$ computation within $n^{1+o(1)}$ time executions. Despite this, computing the approximated attention matrix $U_1 U_2^\top \in \mathbb{R}^{n \times n}$ still necessitates $O(n^2)$ space, leading to significant memory usage. In response to these challenges, we introduce a new algorithm that only reads one pass of the data in a streaming fashion. This method employs sublinear space $o(n)$ to store three sketch matrices, alleviating the need for exact K, V storage. Notably, our algorithm exhibits exceptional memory-efficient performance with super-long tokens. As the token length n increases, our error guarantee diminishes while the memory usage remains nearly constant. This unique attribute underscores the potential of our technique in efficiently handling LLMs in streaming applications.

1 INTRODUCTION

Large Language Models (LLMs) such as ChatGPT (ChatGPT, 2022), InstructGPT (Ouyang et al., 2022), Palm (Chowdhery et al., 2022; Anil et al., 2023), BARD (BARD, 2023), GPT-4 (OpenAI, 2023), LLAMA (Touvron et al., 2023a), LLAMA 2 (Touvron et al., 2023b), Adobe firefly (Adobe, 2023), have revolutionized various aspects of human work. These models have shown remarkable capabilities in dialog systems (Ni et al., 2023; Deng et al., 2023a;b), document summarization (Huang et al., 2023; Ghadimi & Beigy, 2023; Zhang et al., 2023; Krishna et al., 2023), code completion (Zheng et al., 2023; Liu et al., 2023a; Allal et al., 2023), and question-answering (Rogers et al., 2023; Budler et al., 2023; Roy et al., 2023). However, their performance in these applications is often constrained by the context length.

To prepare for the coming of artificial general intelligence (AGI) (Bubeck et al., 2023), one of the crucial bottlenecks for nowadays LLM is about how to handle super long context. In recent OpenAI DevDay (Nov 6, 2023) (Altman, 2023) [1], OpenAI released a new model that is able to support a 128K-long document. In other words, you can feed a 300-page textbook into LLM. This is already quite surprising. However, to finally achieve AGI, we might need to feed some data that is significantly larger than the memory in a model. For example, what if we can't even store the entire x pages of a book in memory when x is super large?

A longer context length allows the LLM to incorporate more information, potentially leading to more accurate and contextually appropriate responses. This increased capacity for information processing can enhance the LLM's understanding, coherence, and contextual reasoning abilities. Therefore, to optimally utilize pretrained LLMs, it's crucial to efficiently and accurately generate long sequences.

[1] OpenAI DevDay, Opening Keynote. `https://www.youtube.com/watch?v=U9mJuUkhUzk`

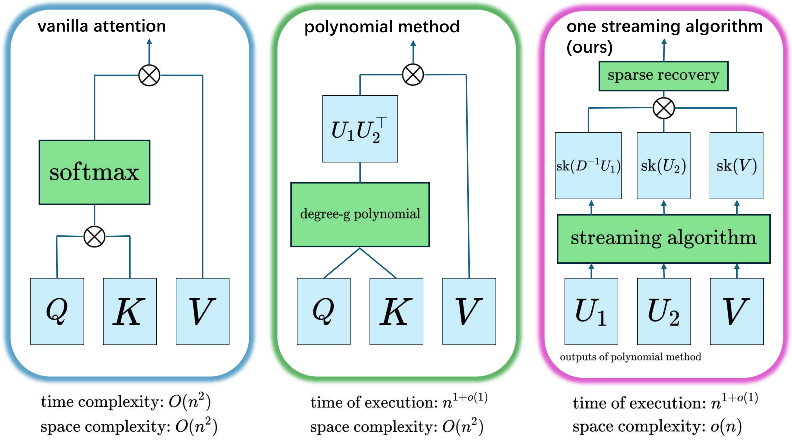

Figure 1: Comparison between our method and previous works. On the left: vanilla attention computation (Vaswani et al., 2017); Middle: fast attention by polynomial method (Alman & Song, 2023a); On the right: one pass algorithm (ours).

Despite the advantages of a long context length, LLMs, especially those based on transformers, face significant computational challenges. Inference with a long context in LLMs is computationally intensive, requiring both $O(n^2)$ space complexity and $O(n^2)$ time complexity to compute the attention output. This computational demand can limit the practical application of LLMs in real-world scenarios, making it a crucial area for further research and optimization.

Previous work (Alman & Song, 2023a;b) has conducted an in-depth study on the fast approximation of attention computation within $n^{1+o(1)}$ time executions without space requirements. Below is a formal definition:

Definition 1.1 (Static Attention Approximation without Space Requirement (Alman & Song, 2023a))**.** *Let $\epsilon \in (0,1)$ denote an accuracy parameter. Given three matrices $Q, K, V \in \mathbb{R}^{n \times d}$, the goal is to construct $T \in \mathbb{R}^{n \times d}$ such that*

$$\|T - \mathsf{Attn}(Q, K, V)\|_\infty \leq \epsilon$$

where

- $\mathsf{Att}(Q, K, V) := D^{-1} A V$
- *$A \in \mathbb{R}^{n \times n}$ is a square matrix $A := \exp(QK^\top / d)$, here we apply $\exp()$ function entry-wisely.*
- *$D \in \mathbb{R}^{n \times n}$ is a diagonal matrix $D := \mathrm{diag}(A \mathbf{1}_n)$ where $\mathbf{1}_n \in \mathbb{R}^n$ is a length-n vector where all the entries are ones.*

However, the memory requirement of caching attention matrix $D^{-1}A$ for LLM's inference is still a considerable issue that consumes $O(n^2)$ space complexity. In this paper, we study the computation-efficiency problem in the context of transformer-based LLMs with super long context. We consider the following problem:

How can we compute the attention with super-long context in space complexity of $o(n)$?

This question is crucial as it directly relates to the computational efficiency of LLMs, particularly when dealing with super-long context lengths. In response to this question, our goal is to develop an effective streaming algorithm. We aim to define and solve the streaming version of approximate attention computation, which is a critical aspect of our research. By addressing this problem, we hope to significantly enhance the computational efficiency of transformer-based LLMs, thereby expanding their applicability in various real-world scenarios. We define the streaming version of approximate attention computation, which is also the problem we aim to solve in this paper:

Definition 1.2 (Streaming Attention Approximation with Sublinear in n Space)**.** *Given $Q, K, V \in \mathbb{R}^{n \times d}$, we're only allowed to use $o(n)$ spaces and read Q, K, V in one pass, and then outputs $T \in \mathbb{R}^{n \times d}$ such that T is close to $D^{-1}AV$.*

1.1 Our Result

In our research, we tackle the challenge of efficiently calculating attention for extremely long input sequences (super-long context) with limited memory resources. Our goal is to process Query Q, Key K, and Value V matrices of size $n \times d$ in a single streaming pass while utilizing only $o(n)$ space.

To address this, we propose a novel one-pass streaming algorithm (Algorithm 1). For $d = O(\log n)$, we first compute low-rank approximation matrices $U_1, U_2 \in \mathbb{R}^{n \times t}$ as in prior work (Alman & Song, 2023a) such that $D^{-1}U_1U_2^\top \approx \mathsf{Attn}(Q, K, V)$.

Next, we introduce sketching matrices $\Phi \in \mathbb{R}^{m_1 \times n}$ (Nakos & Song, 2019), $\Psi \in \mathbb{R}^{m_2 \times n}$ (Alon et al., 1999) to sample U_1, U_2 respectively, where $m_1 = O(\epsilon_1^{-1} k \log n), m_2 = \mathcal{O}(\epsilon_2^{-2} \log n)$ and k controls sparsity.

We present our main result as follows:

Theorem 1.3 (Main Result, informal version of Theorem 4.1)**.** *There is a one pass streaming algorithm (Algorithm 1) that reads $Q, K, V \in \mathbb{R}^{n \times d}$ with $d = O(\log n)$, uses $O(\epsilon_1^{-1} k n^{o(1)} + \epsilon_2^{-2} n^{o(1)})$ spaces and outputs a matrix $T \in \mathbb{R}^{n \times d}$ such that*

- *For each $i \in [d]$, $T_{*,i} \in \mathbb{R}^n$ is $O(k)$-sparse column vector*
- *For each $i \in [d]$, $\|T_i - y_i\|_2 \leq (1+\epsilon) \cdot \min_{k-\mathrm{sparse}\ y'} \|y' - y_i\|_2 + \epsilon_2$ where $y_i = D^{-1}AV_{*,i}$*
- *The succeed probability* 0.99.

The purpose of our work is to address the memory constraints associated with computing attention over very long sequences where the context length $n \gg 2^d$ (potentially infinitely long), then furthermore contribute towards more efficient and scalable transformer models, which could assist in advancing capabilities towards artificial general intelligence (AGI) (Bubeck et al., 2023). Section 2 discusses related work that focuses on approximating attention computation. This includes prior studies on fast approximations without space requirements, which lay the groundwork for our streaming formulation. In Section 3, we outline the preliminary concepts and definitions used in our analysis. This includes problem definition, attention computation, and sketching techniques. Our key technical contributions are presented in Section 4. Here, we introduce a novel one-pass streaming algorithm for attention approximation with sublinear $o(n)$ space complexity. We also state our main theorem, which establishes performance guarantees for our proposed algorithm. Section 4 further provides a detailed proof of the main theorem. This validates that our algorithm is able to process queries, keys and values in a single streaming pass while meeting the stated approximation bounds using limited memory.

2 Related Work

In this section, we briefly review three topics that have close connections to this paper, which are Attention Theory, Streaming Algorithm and Improving LLM's Utilization of Long Text.

Attention Theory Numerous recent studies have explored attention computation in Large Language Models (LLMs) (Kitaev et al., 2020; Tay et al., 2020; Chen et al., 2021; Zandieh et al., 2023; Tarzanagh et al., 2023; Sanford et al., 2023; Panigrahi et al., 2023; Zhang et al., 2020; Arora & Goyal, 2023; Tay et al., 2021; Deng et al., 2023d; Xia et al., 2023; Deng et al., 2023c; Kacham et al., 2023; Alman & Song, 2023a; Brand et al., 2023; Deng et al., 2023e; Gao et al., 2023a; Li et al., 2023c;b; Sinha et al., 2023; Han et al., 2023; Alman & Song, 2023b; Gao et al., 2023b; Alman & Song, 2023a; Han et al., 2023; Kacham et al., 2023; Chu et al., 2023). Some have focused on the benefits of multiple attention heads, showing improved optimization and generalization (Deora et al., 2023). Others have proposed methods like Deja Vu to reduce computational cost during inference without sacrificing quality or learning ability (Liu et al., 2023d). Formal analyses have examined lower and upper bounds for attention computation (Zandieh et al., 2023; Alman & Song, 2023a;b), while dynamic attention computation has also been investigated (Brand et al., 2023). Regression problems within in-context learning for LLMs have been addressed, with a unique approach using matrix formulation (Gao et al., 2023c). These studies collectively contribute to our understanding of attention models and their optimization, generalization, and efficiency.

Streaming Algorithm Streaming algorithms have been extensively studied in graph problems (Kapralov et al., 2014; Assadi et al., 2019a; Farhadi et al., 2020; Bernstein, 2020; Feigenbaum et al., 2004; McGregor, 2005; Paz & Schwartzman, 2017; Ahn & Guha, 2011; Eggert et al., 2012; Goel et al., 2012; Kapralov, 2013; Dobzinski et al., 2014; Ahn & Guha, 2018; Assadi et al., 2020; Assadi & Raz, 2020; Assadi et al., 2021; Ahn & Guha, 2011; 2018; Assadi et al., 2022), spanning trees (Chang et al., 2020), convex programming (Assadi et al., 2019b; Liu et al., 2023c), cardinality estimation (Flajolet et al., 2007), frequency estimation (Alon et al., 1999; Hsu et al., 2019), sampler data structures (Jayaram & Woodruff, 2021), heavy hitter detection (Larsen et al., 2019), and sparse recovery (Nakos & Song, 2019). These studies focus on developing efficient algorithms for various problem domains, such as processing massive graphs, constructing spanning trees, optimizing convex programs, estimating cardinality and frequency, designing sampler data structures, detecting heavy hitters, and recovering sparse signals. The advancements in these areas contribute to the development of efficient and scalable algorithms for real-time analysis of streaming data.

Improving LLMs' Utilization of Long Text Extensive research has been conducted on the application of Large Language Models (LLMs) to lengthy texts (Su et al., 2021; Press et al., 2021; Chen et al., 2023; Dao et al., 2022; Dao, 2023; Zaheer et al., 2020; Beltagy et al., 2020; Wang et al., 2020; Kitaev et al., 2020; Peng et al., 2023). These studies aim to optimize LLMs to effectively capture and utilize the content within longer contexts, rather than treating them solely as inputs. However, despite advancements in these two directions, competent utilization of lengthy contexts within LLMs remains a challenge, as highlighted by recent works (Liu et al., 2023b; Li et al., 2023a).

The effective usage of prolonged contexts poses a significant challenge in the development and application of LLMs. While research has focused on improving LLMs' understanding of longer texts, successfully leveraging this understanding for improved performance is not guaranteed. The challenge lies in effectively incorporating and utilizing the information contained within lengthy contexts, ensuring that LLMs can make accurate and meaningful predictions based on this additional context.

3 Preliminary

Notations. We use $\mathrm{poly}(n)$ to denote $O(n^c)$ where $c \geq 1$ is some constant.

For a vector $x \in \mathbb{R}^n$, we use $\|x\|_2$ to denote its ℓ_2 norm.

We use $\|A\|_\infty$ to denote the ℓ_∞ norm of A, i.e., $\|A\|_\infty := \max_{i,j} |A_{i,j}|$.

We use $\|A\|$ to denote the spectral norm of a matrix. Then it is obvious that $\|A\| \geq \max_j \|A_{*,j}\|_2$.

For a vector $x \in \mathbb{R}^n$, we say x is k-sparse if and only there are k nonzero entries in x.

For a vector $w \in \mathbb{R}^n$, we use $\mathrm{diag}(w) \in \mathbb{R}^{n \times n}$ to denote a diagonal matrix where the i,i-th entry on diagonal is w_i.

We use $\Pr[]$ to denote the probability.

3.1 Polynomial Method

We state a tool from previous work.

Lemma 3.1 (Error Approximation, Lemma 3.6 in (Alman & Song, 2023a)). *if the following conditions*

- *Let* $Q, K, V \in \mathbb{R}^{n \times d}$
- *Let* $d = O(\log n)$*,* $B = O(\sqrt{\log n})$
- *Let* $\|Q\|_\infty, \|K\|_\infty, \|V\|_\infty \leq B$

- *Let* $A := \exp(QK^\top/d)$
- $D := \mathrm{diag}(A\mathbf{1}_n)$

Then, there are matrices $U_1, U_2 \in \mathbb{R}^{n \times t}$ *such that*

- *Part 1.* $t = n^{o(1)}$
- *Part 2. For* i*-th row in* U_1*, we can construct it based on* i*-th row in* Q *in* $O(t+d)$ *time.*
- *Part 3. For* i*-th row in* U_2*, we can construct it based on* i*-th row in* K *in* $O(t+d)$ *time.*
- *Part 4. Let* $\widetilde{A} := U_1 U_2^\top$*, let* $\widetilde{D} := \mathrm{diag}(\widetilde{A}\mathbf{1}_n)$*, then*
$$\|D^{-1}Av - \widetilde{D}^{-1}\widetilde{A}v\|_\infty \leq 1/\operatorname{poly}(n)$$

3.2 Sketching Matrices

Definition 3.2 (k-wise independence)**.** *We say* $\mathcal{H} = \{h : [m] \to [l]\}$ *is a* k*-wise independent hash family if* $\forall i_1 \neq i_2 \neq \cdots \neq i_k \in [n]$ *and* $\forall j_1, \cdots, j_k \in [l]$*,*

$$\Pr_{h \in \mathcal{H}}[h(i_1) = j_1 \wedge \cdots \wedge h(i_k) = j_k] = \frac{1}{l^k}.$$

Definition 3.3 (Random Gaussian matrix)**.** *We say* $\Psi \in \mathbb{R}^{m \times n}$ *is a random Gaussian matrix if all entries are sampled from* $\mathcal{N}(0, 1/m)$ *independently.*

Definition 3.4 (AMS sketch matrix (Alon et al., 1999))**.** *Let* $h_1, h_2, \cdots, h_m$ *be* m *random hash functions picking from a 4-wise independent hash family* $\mathcal{H} = \{h : [n] \to \{-\frac{1}{\sqrt{m}}, +\frac{1}{\sqrt{m}}\}\}$*. Then* $\Psi \in \mathbb{R}^{m \times n}$ *is a AMS sketch matrix if we set* $\Psi_{i,j} = h_i(j)$*.*

Note that in streaming setting, we never need to explicit write the $m \times n$ *matrix. That is too expensive since it takes* $\Omega(n)$ *space. It is well-known that in the streaming area, we only need to store those* m *hash functions, and each hash function only needs* $O(\log n)$*-bits. Thus, the overall store for storing* Φ *is just* $O(m \log n)$ *bits*

3.3 Approximate Matrix Product

Lemma 3.5 (Johnson–Lindenstrauss lemma, folklore, (Johnson & Lindenstrauss, 1984))**.** *Let* $m_2 = O(\epsilon^{-2}\log(1/\delta))$*. For any fixed vectors* u *and* $v \in \mathbb{R}^n$*, let* $\Psi \in \mathbb{R}^{m_2 \times n}$ *denote a random Gaussian/AMS matrix, we have*

$$\Pr[|\langle \Psi u, \Psi v\rangle - \langle u, v\rangle| \leq \epsilon \|u\|_2 \|v\|_2] \geq 1 - \delta$$

Lemma 3.6. *If the following conditions hold*

- *Let* $\delta \in (0,1)$ *denote the failure probability*
- *Let* $\epsilon_2 \in (0,1)$ *denote the accuracy parameter*
- *Let* $m_2 = O(\epsilon_2^{-2}\log(nd/\delta))$*.*
- *Let* $\|V\| \leq 1/\sqrt{n}$*.*

Then we have: with probability $1 - \delta$

- **Part 1.** *for all* $j \in [n], i \in [d]$
$$|(\widetilde{D}^{-1}U_1U_2^\top V)_{j,i} - (\widetilde{D}^{-1}U_1U_2^\top \Psi^\top \Psi V)_{j,i}| \leq \epsilon_2/\sqrt{n}$$
- **Part 2.** *for all* $i \in [d]$*, we have*
$$\|\widetilde{D}^{-1}U_1U_2^\top V_{*,i} - \widetilde{D}^{-1}U_1U_2^\top \Psi^\top \Psi V_{*,i}\|_2 \leq \epsilon_2$$

Proof. First of all, $\|V\| \leq 1/\sqrt{n}$ directly implies that

$$\max_{i \in [d]} \|V_{*,i}\|_2 \leq 1/\sqrt{n} \tag{1}$$

The proof follows from applying Lemma 3.5 and applying a union bound over nd coordinates.

Proof of Part 1. For each $j \in [n]$, for each $i \in [d]$, we can show that

$$\begin{aligned}
& |(\widetilde{D}^{-1} U_1 U_2^\top V)_{j,i} - (\widetilde{D}^{-1} U_1 U_2^\top \Psi^\top \Psi V)_{j,i}| \\
\leq & \epsilon_2 \cdot \|(\widetilde{D}^{-1} U_1 U_2^\top)_{j,*}\|_2 \cdot \|V_{*,i}\|_2 \\
\leq & \epsilon_2 \cdot \|V_{*,i}\|_2 \\
\leq & \epsilon_2 \cdot \frac{1}{\sqrt{n}}
\end{aligned}$$

where the first step follows from Lemma 3.5, the second step follows from $\|(\widetilde{D}^{-1} U_1 U_2^\top)_{j,*}\|_2 \leq \|(\widetilde{D}^{-1} U_1 U_2^\top)_{j,*}\|_1 = 1$, the third step follows from Eq. (1).

Proof of Part 2. For each $i \in [d]$, we can show that

$$\begin{aligned}
& \|\widetilde{D}^{-1} U_1 U_2^\top V_{*,i} - \widetilde{D}^{-1} U_1 U_2^\top \Psi^\top \Psi V_{*,i}\|_2 \\
= & \|(\widetilde{D}^{-1} U_1 U_2^\top V)_{*,i} - (\widetilde{D}^{-1} U_1 U_2^\top \Psi^\top \Psi V)_{*,i}\|_2 \\
\leq & (n \cdot (\epsilon_2/\sqrt{n})^2)^{1/2} \\
\leq & \epsilon_2
\end{aligned}$$

where the first step follows from $AB_{*,i} = (AB)_{*,i}$ for all $i \in [d]$, second step follows from Part 1, the third step follows from definition of ℓ_2 norm. □

3.4 SPARSE RECOVERY

We state a sparse recovery tool from previous work (Nakos & Song, 2019).

Lemma 3.7 (Sparse Recovery, Theorem 1.1 in (Nakos & Song, 2019))**.** *For any vector $x \in \mathbb{R}^n$, there is an oblivious sketching matrices $\Phi \in \mathbb{R}^{m_1 \times n}$ such that*

- *Let k denote a positive integer.*
- $m_1 = O(\epsilon_1^{-1} k \log n)$
- *The encoding/update time(or the column sparsity of Φ) is $O(\log n)$*
  - *In particular, computing $\Phi e_i \Delta$ takes $O(\log n)$ for any scalar $\Delta \in \mathbb{R}$, and one-hot vector $e_i \in \mathbb{R}^n$.*
  - *For convenient of later analysis, we use $z = \Phi x$.*
  - *The space is to store Φ is $O(m)$ bits*
- *The decoding/recover time is $O(m_1 \log n)$*
- *The algorithm is able to output a k-sparse vector $x' \in \mathbb{R}^n$ such that*
$$\|x' - x\|_2 \leq (1 + \epsilon_1) \min_{k-\mathrm{sparse}\ x_k} \|x_k - x\|_2$$
- *The succeed probability is* 0.999

4 ANALYSIS

We present the main result of this paper.

Theorem 4.1 (Main Result, formal version of Theorem 1.3)**.** *If the following conditions hold*

- *Let $d = O(\log n)$*

Algorithm 1 Our One-Pass Streaming Algorithm for matrices $Q, K \in \mathbb{R}^{n \times d}$ and $V \in \mathbb{R}^{n \times d}$. The goal of this algorithm is to provide a k-sparse approximation to column of $Y = D^{-1} \exp(QK^\top)V \in \mathbb{R}^{n \times d}$.

procedure MAINALGORITHM($Q \in \mathbb{R}^{n \times d}, K \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{n \times d}$) $\triangleright$ Theorem 4.1
  /*Create $O((m_2 + m_1) \times t)$ spaces*/
  Let $\mathrm{sk}(U_2) \in \mathbb{R}^{m_2 \times t}$ denote the sketch of U_2 $\triangleright$ After stream, we will have $\mathrm{sk}(U_2) = \Psi U_2$
  Let $\mathrm{sk}(V) \in \mathbb{R}^{m_2 \times d}$ denote the sketch of V $\triangleright$ $\mathrm{sk}(V) = \Psi V$
  Let $\mathrm{sk}(D^{-1}U_1) \in \mathbb{R}^{m_1 \times t}$ denote the sketch of $D^{-1}U_1$ $\triangleright$ $\mathrm{sk}(D^{-1}U_1) = \Phi D^{-1} U_1$
  Let $\mathrm{prod}(U_2^\top \mathbf{1}_n) \in \mathbb{R}^t$ denote the $U_2^\top \mathbf{1}_n$
  /* Initialization */
  We initialize all the matrices/vector objects to be zero
  $\mathrm{sk}(U_2) \leftarrow \mathbf{0}_{m_2 \times t}$, $\mathrm{sk}(V) \leftarrow \mathbf{0}_{m_2 \times d}$, $\mathrm{sk}(D^{-1}U_1) \leftarrow \mathbf{0}_{m_1 \times t}$, $\mathrm{prod}(U_2^\top \mathbf{1}_n) \leftarrow \mathbf{0}_t$
  /*Read V in streaming and compute sketch of V*/
  Read V in one pass stream, and compute $\mathrm{sk}(V) = \Psi V$
  $\triangleright$ We will have $\mathrm{sk}(V) = \Psi V$ when we reach this line
  /*Read K in streaming and compute sketch of U_2*/
  for $i = 1 \to n$ **do**
    Read one row of $K \in \mathbb{R}^{n \times d}$
    $\triangleright$ We construct U_2 according to Lemma 3.1
    We construct one row of $U_2 \in \mathbb{R}^{n \times t}$, let us call that row to be $(U_2)_{i,*}$ which has length t
    $\mathrm{prod}(U_2^\top \mathbf{1}_n) \leftarrow \mathrm{prod}(U_2^\top \mathbf{1}_n) + ((U_2)_{i,*})^\top$
    $\mathrm{sk}(U_2) \leftarrow \mathrm{sk}(U_2) + \underbrace{\Psi}_{m_2 \times n} \underbrace{e_i}_{n \times 1} \underbrace{(U_2)_{i,*}}_{1 \times t}$
  end for
  $\triangleright$ We will have $\mathrm{prod}(U_2^\top \mathbf{1}_n) = U_2^\top \mathbf{1}_n$ when reach this line
  $\triangleright$ We will have $\mathrm{sk}(U_2) = \Psi U_2$ when reach this line
  /* Read Q in streaming and compute sketch of $D^{-1}U_1$*/
  for $i = 1 \to n$ **do**
    Read one row of $Q \in \mathbb{R}^{n \times d}$
    $\triangleright$ We construct U_1 according to Lemma 3.1
    We construct one row of $U_1 \in \mathbb{R}^{n \times t}$, let us call that row to be $(U_1)_{i,*}$ which has length t
    Compute $D_{i,i} \leftarrow \underbrace{(U_1)_{i,*}}_{1 \times t} \underbrace{\mathrm{prod}(U_2^\top \mathbf{1}_n)}_{t \times 1}$
    $\mathrm{sk}(D^{-1}U_1) \leftarrow \mathrm{sk}(D^{-1}U_1) + \underbrace{\Phi}_{m_1 \times n} \underbrace{e_i}_{n \times 1} \underbrace{D_{i,i}^{-1}(U_1)_{i,*}}_{1 \times t}$
  end for
  $\triangleright$ We will have $\mathrm{sk}(D^{-1}U_1) = \Phi D^{-1} U_1$ when we reach this line
  /* Run Sparse Recovery Algorithm */
  Compute $Z \leftarrow \mathrm{sk}(D^{-1}U_1)\, \mathrm{sk}(U_2)^\top \mathrm{sk}(V)$ $\triangleright$ $Z \in \mathbb{R}^{m_1 \times d}$
  Run sparse recovery on each column of $Z \in \mathbb{R}^{m_1 \times d}$ to get approximation to the corresponding column of $Y \in \mathbb{R}^{n \times d}$
end procedure

- *Let* $B = O(\sqrt{\log n})$
- *Let* $\|Q\|_\infty \leq B$, $\|K\|_\infty \leq B$, $\|V\| \leq 1/\sqrt{n}$
- *Let* $A := \exp(QK^\top/d) \in \mathbb{R}^{n \times n}$
- *Let* $D := \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n \times n}$

There is a one pass streaming algorithm (Algorithm 1) that reads $Q, K, V \in \mathbb{R}^{n \times d}$ *uses*

$$O(\epsilon_1^{-1} k n^{o(1)} + \epsilon_2^{-2} n^{o(1)})$$

spaces and outputs a matrix $T \in \mathbb{R}^{n \times d}$ *such that*

- *For each* $i \in [d]$, $T_{*,i} \in \mathbb{R}^n$ *is* $O(k)$*-sparse column vector*

- *For each* $i \in [d]$, $\|T_i - y_i\|_2 \leq (1+\epsilon_1) \cdot \min_{k-\mathrm{sparse}\ y'} \|y' - y_i\|_2 + \epsilon_2$ *where* $y_i = D^{-1} A V_{*,i}$
- *The succeed probability* 0.99*.*
- *The decoding time is* $O(\epsilon_1^{-1} k n^{o(1)})$*.*

Proof. The streaming will be able to construct sketch $Z \in \mathbb{R}^{m_1 \times d}$ which is

$$\begin{aligned} Z = & \ \mathrm{sk}(D^{-1} U_1) \, \mathrm{sk}(U_2)^\top \, \mathrm{sk}(V) \\ = & \ \Phi D^{-1} U_1 U_2 \Psi^\top \Psi V \end{aligned}$$

Running Lemma 3.7 on Z is essentially, doing sparse recovery for $D^{-1} U_1 U_2 \Psi^\top \Psi V$.

Since $D^{-1} U_1 U_2 \Psi^\top \Psi V$ is close to $D^{-1} U_1 U_2 V$, thus, we can finally show the error guarantees for $D^{-1} U_1 U_2 V$.

Proof of Space Requirement.

From the algorithm it is easy to see, the space is coming from two parts

- $O(m_1 t)$ spaces for object $\mathrm{sk}(D^{-1} U_1)$
- $O(m_2 t)$ spaces for object $\mathrm{sk}(U_2)$

From Lemma 3.1, we have

$$t = n^{o(1)}$$

From Lemma 3.6, we have

$$m_2 = O(\epsilon_2^{-2} \log(n))$$

From Lemma 3.7, we have

$$m_1 = O(\epsilon_1^{-1} k \log n)$$

Thus, total space is

$$O(m_1 t + m_2 t) = O(\epsilon_1^{-1} k n^{o(1)} + \epsilon_2^{-2} n^{o(1)}).$$

Proof of Decoding Time.

The decoding time is directly following from Lemma 3.7, it is

$$O(m_1 \log n) = O(\epsilon_1^{-1} k n^{o(1)} \log n) = O(\epsilon_1^{-1} k n^{o(1)}).$$

where the first step follows from choice of m_1, the last step follows from $O(\log n) = O(n^{o(1)})$.

Proof of Error Guarantees.

To finish the proofs, we define a list of variables

- $y_i = D^{-1} A V_{*,i} \in \mathbb{R}^n$
- $\widetilde{y}_i = \widetilde{D}^{-1} \widetilde{A} V_{*,i} \in \mathbb{R}^n$
- $\widehat{y}_i = \widetilde{D}^{-1} \widetilde{A} \Psi^\top \Psi V_{*,i} \in \mathbb{R}^n$
- Let ξ_1 be the value that $\|y_i - \widetilde{y}_i\|_2 \leq \xi_1$ ($\xi_1 = 1/\mathrm{poly}(n)$, by Part 4 of Lemma 3.1)
- Let ξ_1 be the value that $\|\widetilde{y}_i - \widehat{y}_i\|_2 \leq \xi_2$ ($\xi_2 = \epsilon_2$, by Part 2 of Lemma 3.6)

Firstly, we can show that

$$\begin{aligned}\|y_i - \widehat{y}_i\|_2 &\le \|y_i - \widetilde{y}_i\|_2 + \|\widetilde{y}_i - \widehat{y}\|_2 \\ &\le \xi_1 + \xi_2 \end{aligned} \tag{2}$$

We can show

$$\begin{aligned}\|T_i - y_i\|_2 &\le \|T_i - \widehat{y}_i\|_2 + \|\widehat{y}_i - y_i\|_2 \\ &\le \|T_i - \widehat{y}_i\|_2 + \xi_1 + \xi_2 \\ &\le (1+\epsilon_1) \min_{k-\mathrm{sparse}\ y'} \|y' - \widehat{y}_i\|_2 + \xi_1 + \xi_2 \\ &\le (1+\epsilon_1) \min_{k-\mathrm{sparse}\ y'} \|y' - y_i\|_2 \\ &\quad + (1+\epsilon)(\xi_1 + \xi_2) + (\xi_1 + \xi_2) \\ &\le (1+\epsilon_1) \min_{k-\mathrm{sparse}\ y'} \|y' - y_i\|_2 + 3(\xi_1 + \xi_2) \\ &\le (1+\epsilon_1) \min_{k-\mathrm{sparse}\ y'} \|y' - y_i\|_2 + O(\epsilon_2)\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from Eq. (2), the third step follows from Lemma 3.7, the fourth step follows from triangle inequality, the fifth step follows from $\epsilon_1 \in (0, 1)$ and the last step follows from $\xi_1 = 1/\operatorname{poly}(n) < \xi_2 = \epsilon_2$.

Proof of Failure Probability.

The failure probability of Lemma 3.6 is $\delta = 1/\operatorname{poly}(n)$. The failure probability of Lemma 3.7 is 0.001. Taking a union bound over those Lemmas, we get failure probability 0.01 here.

In particular, the failure probability is at most

$$\begin{aligned}0.001 + 1/\operatorname{poly}(n) &\le 0.001 + 0.001 \\ &\le 0.01.\end{aligned}$$

Thus, we complete the proof. □

4.1 A General Result

We state a result for solving cross attention ($X_1 \neq X_2$). Using our framework to solve self-attention ($X_1 = X_2$), then the algorithm will need two passes, instead of one pass.

Corollary 4.2 (An application of Theorem 4.1)**.** *If the following conditions hold*

- *Let* $d = O(\log n)$, $B = O(\sqrt{\log n})$
- *Let* $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$
- *Let* $X_1, X_2 \in \mathbb{R}^{n \times d}$
- *Let* $Q = X_1 W_Q \in \mathbb{R}^{n \times d}$, $K = X_2 W_K \in \mathbb{R}^{n \times d}$, $V = X_2 W_V \in \mathbb{R}^{n \times d}$
- *Let* $\|Q\|_\infty \le B$, $\|K\|_\infty \le B$, $\|V\| \le 1/\sqrt{n}$
- *Let* $A := \exp(QK^\top / d) \in \mathbb{R}^{n \times n}$
- *Let* $D := \operatorname{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n \times n}$

There is a one pass streaming algorithm (Algorithm 2) that reads $X \in \mathbb{R}^{n \times d}$, $W_Q, W_K, W_V \in \mathbb{R}^{n \times d}$ *uses*

$$O(\epsilon_1^{-1} k n^{o(1)} + \epsilon_2^{-2} n^{o(1)})$$

spaces and outputs a matrix $T \in \mathbb{R}^{n \times d}$ *such that*

- *For each* $i \in [d]$, $T_{*,i} \in \mathbb{R}^n$ *is* $O(k)$*-sparse column vector*

- *For each* $i \in [d]$, $\|T_i - y_i\|_2 \leq (1+\epsilon_1) \cdot \min_{k-\text{sparse } y'} \|y' - y_i\|_2 + \epsilon_2$ *where* $y_i = D^{-1} A V_{*,i}$
- *The succeed probability* 0.99*.*
- *The decoding time is* $O(\epsilon_1^{-1} k n^{o(1)})$*.*

Proof. The proofs are similar to Theorem 4.1. The only difference between streaming algorithms (Algorithm 1 and Algorithm 2) is that, in Algorithm 2 we don't receive each row of Q (similarly as K, V) on the fly anymore. Instead, we store weight W_Q, and we receive each row of X_1 on the fly. Whenever we see a row of X_1, we will compute matrix vector multiplication for that row and weight W_Q.

Similarly, we applied the same strategy for K and V. □

REFERENCES

Adobe. Adobe firefly. *https://www.adobe.com/sensei/generative-ai/firefly.html*, 2023.

Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *International Colloquium on Automata, Languages, and Programming*, pp. 526–538. Springer, 2011.

Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Transactions on Parallel Computing (TOPC)*, 4(4):1–40, 2018.

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, et al. Santacoder: don't reach for the stars! *arXiv preprint arXiv:2301.03988*, 2023.

Josh Alman and Zhao Song. Fast attention requires bounded entries. In *NeurIPS*, 2023a.

Josh Alman and Zhao Song. How to capture higher-order correlations? generalizing matrix softmax attention to kronecker computation. *arXiv preprint arXiv:2310.04064*, 2023b.

Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

Sam Altman. Openai devday. *https://www.youtube.com/watch?v=U9mJuUkhUzk*, 2023.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Sanjeev Arora and Anirudh Goyal. A theory for emergence of complex skills in language models. *arXiv preprint arXiv:2307.15936*, 2023.

Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 342–353. IEEE, 2020.

Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. Coresets meet edcs: algorithms for matching and vertex cover on massive graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1616–1635. SIAM, 2019a.

Sepehr Assadi, Nikolai Karpov, and Qin Zhang. Distributed and streaming linear programming in low dimensions. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pp. 236–253, 2019b.

Sepehr Assadi, Gillat Kol, Raghuvansh R Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 354–364. IEEE, 2020.

Sepehr Assadi, S Cliff Liu, and Robert E Tarjan. An auction algorithm for bipartite matching in streaming and massively parallel computation models. In *Symposium on Simplicity in Algorithms (SOSA)*, pp. 165–171. SIAM, 2021.

Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In *SODA*. arXiv preprint arXiv:2011.03495, 2022.

BARD. Try bard, an ai experiment by google. *Google*, February 2023. URL `https://bard.google.com/`.

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

Aaron Bernstein. Improved bound for matching in random-order streams. *arXiv preprint arXiv:2005.00417*, 2020.

Jan van den Brand, Zhao Song, and Tianyi Zhou. Algorithm and hardness for dynamic attention maintenance in large language models. *arXiv preprint arXiv:2304.02207*, 2023.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Leona Cilar Budler, Lucija Gosak, and Gregor Stiglic. Review of artificial intelligence-based question-answering systems in healthcare. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1487, 2023.

Yi-Jun Chang, Martin Farach-Colton, Tsan-Sheng Hsu, and Meng-Tsung Tsai. Streaming complexity of spanning tree computation. In *37th international symposium on theoretical aspects of computer science (STACS)*, 2020.

ChatGPT. Optimizing language models for dialogue. *OpenAI Blog*, November 2022. URL `https://openai.com/blog/chatgpt/`.

Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. Mongoose: A learnable lsh framework for efficient neural network training. In *International Conference on Learning Representations*, 2021.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Timothy Chu, Zhao Song, and Chiwun Yang. How to protect copyright data in optimization of large language models? *arXiv preprint arXiv:2308.12247*, 2023.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Jiawen Deng, Hao Sun, Zhexin Zhang, Jiale Cheng, and Minlie Huang. Recent advances towards safe, responsible, and moral dialogue systems: A survey. *arXiv preprint arXiv:2302.09270*, 2023a.

Yang Deng, Wenqiang Lei, Wai Lam, and Tat-Seng Chua. A survey on proactive dialogue systems: Problems, methods, and prospects. *arXiv preprint arXiv:2305.02750*, 2023b.

Yichuan Deng, Yeqi Gao, and Zhao Song. Solving tensor low cycle rank approximation. In *BigData*. arXiv preprint arXiv:2304.06594, 2023c.

Yichuan Deng, Zhihang Li, and Zhao Song. Attention scheme inspired softmax regression. *arXiv preprint arXiv:2304.10411*, 2023d.

Yichuan Deng, Sridhar Mahadevan, and Zhao Song. Randomized and deterministic attention sparsification algorithms for over-parameterized feature dimension. *arxiv preprint: arxiv 2304.03426*, 2023e.

Puneesh Deora, Rouzbeh Ghaderi, Hossein Taheri, and Christos Thrampoulidis. On the optimization and generalization of multi-head attention. *arXiv preprint arXiv:2310.12680*, 2023.

Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 233–242, 2014.

Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012.

Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mah, Anup Rao, and Ryan A Rossi. Approximate maximum matching in random streams. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1773–1785. SIAM, 2020.

Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In *International Colloquium on Automata, Languages, and Programming*, pp. 531–543. Springer, 2004.

Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete mathematics & theoretical computer science*, (Proceedings), 2007.

Yeqi Gao, Sridhar Mahadevan, and Zhao Song. An over-parameterized exponential regression. *arXiv preprint arXiv:2303.16504*, 2023a.

Yeqi Gao, Zhao Song, Weixin Wang, and Junze Yin. A fast optimization view: Reformulating single layer attention in llm based on tensor and svm trick, and solving it in matrix multiplication time. *arXiv preprint arXiv:2309.07418*, 2023b.

Yeqi Gao, Zhao Song, and Shenghao Xie. In-context learning for attention scheme: from single softmax regression to multiple softmax regression via a tensor trick. *arXiv preprint arXiv:2307.02419*, 2023c.

Alireza Ghadimi and Hamid Beigy. Sgcsumm: An extractive multi-document summarization method based on pre-trained language model, submodularity, and graph convolutional neural networks. *Expert Systems with Applications*, 215:119308, 2023.

Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pp. 468–485. SIAM, 2012.

Insu Han, Rajesh Jarayam, Amin Karbasi, Vahab Mirrokni, David P Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. *arXiv preprint arXiv:2310.05869*, 2023.

Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*, 2019.

Kung-Hsiang Huang, Philippe Laban, Alexander R Fabbri, Prafulla Kumar Choubey, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. Embrace divergence for richer insights: A multi-document summarization benchmark and a case study on summarizing diverse information from news articles. *arXiv preprint arXiv:2309.09369*, 2023.

Rajesh Jayaram and David Woodruff. Perfect l_p sampling in a data stream. *SIAM Journal on Computing*, 50(2):382–439, 2021.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. Polysketchformer: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655*, 2023.

Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1679–1697. SIAM, 2013.

Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 734–751. SIAM, 2014.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kalpesh Krishna, Erin Bransom, Bailey Kuehl, Mohit Iyyer, Pradeep Dasigi, Arman Cohan, and Kyle Lo. Longeval: Guidelines for human evaluation of faithfulness in long-form summarization. *arXiv preprint arXiv:2301.13298*, 2023.

Kasper Green Larsen, Jelani Nelson, Huy L Nguyen, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. *Communications of the ACM*, 62(8):95–100, 2019.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can opensource llms truly promise on context length, 2023a.

Yuchen Li, Yuanzhi Li, and Andrej Risteski. How do transformers learn topic structure: Towards a mechanistic understanding. *arXiv preprint arXiv:2303.04245*, 2023b.

Zhihang Li, Zhao Song, and Tianyi Zhou. Solving regularized exp, cosh and sinh regression problems. *arXiv preprint, 2303.15725*, 2023c.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*, 2023a.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023b.

S Cliff Liu, Zhao Song, Hengjie Zhang, Lichen Zhang, and Tianyi Zhou. Space-efficient interior point method, with applications to linear programming and maximum weight bipartite matching. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 88:1–88:14, 2023c.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023d.

Andrew McGregor. Finding graph matchings in data streams. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pp. 170–181. Springer, 2005.

Vasileios Nakos and Zhao Song. Stronger l2/l2 compressed sensing; without iterating. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 289–297, 2019.

Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, and Erik Cambria. Recent advances in deep learning based dialogue systems: A systematic survey. *Artificial intelligence review*, 56(4): 3055–3155, 2023.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Abhishek Panigrahi, Sadhika Malladi, Mengzhou Xia, and Sanjeev Arora. Trainable transformer in transformer. *arXiv preprint arXiv:2307.01189*, 2023.

Ami Paz and Gregory Schwartzman. A (2+ϵ)-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2153–2161. SIAM, 2017.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Anna Rogers, Matt Gardner, and Isabelle Augenstein. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *ACM Computing Surveys*, 55(10): 1–45, 2023.

Pradeep Kumar Roy, Sunil Saumya, Jyoti Prakash Singh, Snehasish Banerjee, and Adnan Gutub. Analysis of community question-answering issues via machine learning and deep learning: State-of-the-art review. *CAAI Transactions on Intelligence Technology*, 8(1):95–117, 2023.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. *arXiv preprint arXiv:2306.02896*, 2023.

Ritwik Sinha, Zhao Song, and Tianyi Zhou. A mathematical abstraction for balancing the trade-off between creativity and reality in large language models. *arXiv preprint arXiv:2306.02295*, 2023.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Davoud Ataee Tarzanagh, Yingcong Li, Christos Thrampoulidis, and Samet Oymak. Transformers as support vector machines. *arXiv preprint arXiv:2308.16898*, 2023.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International conference on machine learning*, pp. 10183–10192. PMLR, 2021.

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

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

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.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

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

Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. Kdeformer: Accelerating transformers via kernel density estimation. *arXiv preprint arXiv:2302.02451*, 2023.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. Extractive summarization via chatgpt for faithful summary generation. *arXiv preprint arXiv:2304.04193*, 2023.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, et al. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x. *arXiv preprint arXiv:2303.17568*, 2023.

A LIMITATIONS

In our work, we propose a single-pass streaming algorithm for the computation of very long sequences of attention, but we recognize some limitations. Limitations relate to the algorithm's reliance on specific assumptions, limitations of the test scope, sensitivity to input quality and data characteristics, and changes in performance as the data size increases.

B SOCIETAL IMPACT

In this paper, we introduce an innovative single-pass algorithm, which can achieve efficient approximation of ultra-long sequence attention computing under sublinear space complexity, and solve the problem of high time and space complexity in current attention computing. Our paper is purely theoretical and empirical in nature (mathematics problem) and thus we foresee no immediate negative ethical impact.

By constructing a specific matrix to approximate the attention output, the algorithm only needs one data traversal and uses sublinear space to store three summary matrices, which greatly reduces the memory requirement. It is especially suitable for processing extremely long sequences. As the sequence length increases, the error is guaranteed to decrease while the memory usage is almost constant, showing excellent memory efficiency when streaming super long tokens.

C ALGORITHM FOR GENERAL RESULT

Here, we state our algorithm for general result in Section 4.1.

Algorithm 2 Our Streaming Algorithm for matrices $X_1 \in \mathbb{R}^{n \times d}$, $X_2 \in \mathbb{R}^{n \times d}$, $W_Q, W_K \in \mathbb{R}^{d \times d}$ and $W_V \in \mathbb{R}^{d \times d}$. The goal of this algorithm is to provide a k-sparse approximation to column of $Y = D^{-1} \exp(QK^\top) V \in \mathbb{R}^{n \times d}$.

procedure MAINALGORITHM($X_1 \in \mathbb{R}^{n \times d}, X_2 \in \mathbb{R}^{n \times d}, W_Q \in \mathbb{R}^{d \times d}, W_K \in \mathbb{R}^{d \times d}, W_V \in \mathbb{R}^{d \times d}$) ▷ Corollary 4.2
  /*Create $O((m_2 + m_1) \times t) + O(d^2)$ spaces*/
  Let $\mathrm{sk}(U_2) \in \mathbb{R}^{m_2 \times t}$ denote the sketch of U_2 ▷ After stream, we will have $\mathrm{sk}(U_2) = \Psi U_2$
  Let $\mathrm{sk}(V) \in \mathbb{R}^{m_2 \times d}$ denote the sketch of V ▷ $\mathrm{sk}(V) = \Psi V$
  Let $\mathrm{sk}(D^{-1}U_1) \in \mathbb{R}^{m_1 \times t}$ denote the sketch of $D^{-1}U_1$ ▷ $\mathrm{sk}(D^{-1}U_1) = \Phi D^{-1} U_1$
  Let $\mathrm{prod}(U_2^\top \mathbf{1}_n) \in \mathbb{R}^t$ denote the $U_2^\top \mathbf{1}_n$
  /* Initialization */
  We initialize all the matrices/vector objects to be zero
  $\mathrm{sk}(U_2) \leftarrow \mathbf{0}_{m_2 \times t}$, $\mathrm{sk}(V) \leftarrow \mathbf{0}_{m_2 \times d}$, $\mathrm{sk}(D^{-1}U_1) \leftarrow \mathbf{0}_{m_1 \times t}$, $\mathrm{prod}(U_2^\top \mathbf{1}_n) \leftarrow \mathbf{0}_t$
  /*Read X_2 in streaming and compute sketch of U_2 and sketch of V*/
  for $i = 1 \to n$ **do**
    Read one row of $X_2 \in \mathbb{R}^{n \times d}$
    We can obtain one row of K and also one row of V (by computing matrix vector multiplication between one row of X_1 and W_K, and X_1 and W_V)
    ▷ We construct U_2 according to Lemma 3.1
    We construct one row of $U_2 \in \mathbb{R}^{n \times t}$, let us call that row to be $(U_2)_{i,*}$ which has length t
    $\mathrm{prod}(U_2^\top \mathbf{1}_n) \leftarrow \mathrm{prod}(U_2^\top \mathbf{1}_n) + ((U_2)_{i,*})^\top$
    $\mathrm{sk}(U_2) \leftarrow \mathrm{sk}(U_2) + \underbrace{\Psi}_{m_2 \times n} \underbrace{e_i}_{n \times 1} \underbrace{(U_2)_{i,*}}_{1 \times t}$
    $\mathrm{sk}(V) \leftarrow \mathrm{sk}(V) + \underbrace{\Psi}_{m_2 \times n} \underbrace{e_i}_{n \times 1} \underbrace{(V_2)_{i,*}}_{1 \times d}$
  end for
  ▷ We will have $\mathrm{prod}(U_2^\top \mathbf{1}_n) = U_2^\top \mathbf{1}_n$ when reach this line
  ▷ We will have $\mathrm{sk}(U_2) = \Psi U_2$ when reach this line
  ▷ We will have $\mathrm{sk}(V) = \Psi V$ when reach this line
  /* Read X_1 in streaming and compute sketch of $D^{-1}U_1$*/
  for $i = 1 \to n$ **do**
    Read one row of $X_1 \in \mathbb{R}^{n \times d}$
    We can obtain one row of Q (by computing matrix vector multiplication between one row of X_1 and W_Q)
    ▷ We construct U_1 according to Lemma 3.1
    We construct one row of $U_1 \in \mathbb{R}^{n \times t}$, let us call that row to be $(U_1)_{i,*}$ which has length t
    Compute $D_{i,i} \leftarrow \underbrace{(U_1)_{i,*}}_{1 \times t} \underbrace{\mathrm{prod}(U_2^\top \mathbf{1}_n)}_{t \times 1}$
    $\mathrm{sk}(D^{-1}U_1) \leftarrow \mathrm{sk}(D^{-1}U_1) + \underbrace{\Phi}_{m_1 \times n} \underbrace{e_i}_{n \times 1} \underbrace{D_{i,i}^{-1}(U_1)_{i,*}}_{1 \times t}$
  end for
  ▷ We will have $\mathrm{sk}(D^{-1}U_1) = \Phi D^{-1} U_1$ when we reach this line
  /* Run Sparse Recovery Algorithm */
  Compute $Z \leftarrow \mathrm{sk}(D^{-1}U_1)\, \mathrm{sk}(U_2)^\top \, \mathrm{sk}(V)$ ▷ $Z \in \mathbb{R}^{m_1 \times d}$
  Run sparse recovery on each column of $Z \in \mathbb{R}^{m_1 \times d}$ to get approximation to the corresponding column of $Y \in \mathbb{R}^{n \times d}$
end procedure