Dean&Francis

ISSN 2959-6157

Enhancing Large Language Model Performance through Sketching Techniques

Junyin Zhang[1,*],

Zecen Ding[2],

Yitian Wan[3],

Yuheng Shen[4]

[1]Department of Mathematics, The University of British Columbia, Vancouver, V6T 1Z4, Canada, junyinsuc2021@126.com
[2]Department of Statistical Sciences, University of Toronto, Toronto, M5G 1X6, Canada, zecen.ding@mail.utoronto.ca
[3]Shanghai Shangde Experimental School, Shanghai, China, Qinglian67@163.com
[4]Morrissey College of Arts & Sciences, Boston College, Boston, 02467, The USA, shenby@bc.edu

Abstract:
This article explores the theoretical application of sketching techniques to Large Language Models (LLMs), which use deep learning and extensive datasets for natural language processing tasks. The study summarizes two approaches: PolySketchFormer and Prompt Sketching. PolySketchFormer accelerates transformer models using sketching for performance optimization, while Prompt Sketching aims to enhance model accuracy. The article delves into the theory and processes of these methods, highlighting their advantages and potential implications for advancing LLM capabilities.

Keywords: Large Language Model (LLM), Sketching, PolySketchFormer, Prompt Sketching

1. Introduction

Large Language Models (LLMs) represent a significant advancement in artificial intelligence, employing deep learning techniques and incorporating a vast array of parameters to understand and process human language, which empowers them to execute a broad spectrum of natural language processing tasks, including but not limited to text generation, translation, summarization, and other related activities. The foundation of LLMs often includes architectures such as the transformer, which utilizes mechanisms like self-attention to assign varying levels of importance to different words within a sequence. This approach enables the models to achieve a nuanced and context-aware comprehension and generation of language. “LLMs have emerged as cutting-edge artificial intelligence systems that can process and generate text with coherent communication and generalize to multiple tasks.” [1].

Sketching typically refers to an algorithmic technique designed to expedite large-scale computations. The core concept involves data compression by multiplying a large matrix by a smaller, often random matrix with specific properties, thereby reducing computational demands. Linear sketching operates

ISSN 2959-6157

on the principle that "given a matrix, one first compresses it to a much smaller matrix by multiplying it by a (usually) random matrix with certain properties" [2].

In this article, we aim to theoretically examine the application of sketching techniques to LLMs, proposing various approaches and discussing their potential implications. Our research has focused on PolySketchFormer, which employs sketching to accelerate transformer models and enhance their performance [3], as well as Prompt Sketching, in which the process is template creation, variable prediction, and intermediate instruction insertion. The Prompt Sketching seeks to improve model accuracy [4]. This article delineates the theoretical framework and processes underlying each method and their advantages. We integrate the Polysketchformer and Prompt sketching technology and analyze possible improvements to find out the possibility of using sketching to improve LLM's performance.

2. Methodology

The origins of the scholarly material for this research encompass the institutional library, the Google Scholar digital repository, and the esteemed suggestions offered by the professor. Initially, we intended to delve into the practical applications of sketching methodologies and the Retrieval-Augmented Generation (RAG) framework. However, our investigative trajectory ultimately gravitated toward the Large Language Models (LLMs), with a particular interest in examining the potential integration of sketching techniques within the context of LLMs.

Through a meticulous process of literature review, we meticulously evaluated the available sources, engaged in an in-depth analysis of the pertinent data and scholarly insights, and meticulously organized the findings. This rigorous methodology culminated in the synthesis of a comprehensive summary, which serves as the foundation for the present academic discourse.

3. Literature Review

3.1 Theme 1

Large Language Models (LLMs) have significantly advanced natural language processing (NLP) by effectively understanding and generating human language [5,6]. However, their substantial computational cost and memory requirements pose notable challenges [7]. Sketching techniques have emerged as a promising solution to mitigate these issues by providing efficient approximations of data representations and operations [2].

One notable advancement in this area is the PolySketchFormer, which integrates polynomial kernel methods with sketching techniques to accelerate Transformer models. The primary challenge that PolySketchFormer addresses is the computational complexity associated with the self-attention mechanism in Transformers, which is $O\left(n^2\right)$ for a sequence length n [3]. PolySketchFormer employs polynomial kernels to approximate the self-attention scores, thereby reducing this complexity. The polynomial kernel function, $K\left(x,y\right)=(x^T y+c)^d$, allows for a low-dimensional representation of the attention matrix, enabling faster computation [3].

In addition to polynomial kernels, PolySketchFormer incorporates sketching techniques to further optimize performance. Sketching reduces the dimensionality of the data by creating compact summaries or sketches that approximate the original data with high probability. Specifically, a method known as Polynomial Sketching is used to maintain the frequency counts of elements in the data stream, significantly lowering the memory footprint and computational load [3]. This combination allows PolySketchFormer to balance the trade-off between computational efficiency and model accuracy.

PolySketchFormer achieves these improvements through several key innovations:

1. Polynomial Attention: By using high-degree polynomial kernels, the model replaces the softmax function in the attention mechanism without sacrificing model quality [3].
2. Approximate Feature Mapping: The model applies sketching techniques from numerical linear algebra to compute low-dimensional approximate feature mappings that approximate the high-dimensional polynomial kernel features. This approach ensures that the attention mechanism remains computationally feasible while maintaining performance [3].
3. Handling Causal Masks: PolySketchFormer uses a block-based algorithm to efficiently apply causal masking, which is crucial for autoregressive tasks like language modeling [3]. This method allows the model to maintain linear-time complexity while handling the dependencies within long sequences.

These innovations are empirically validated by training language models on datasets like PG19, Wikipedia, and C4, using Google Cloud TPUs. The results demonstrate that PolySketchFormer achieves a 2x speedup in training for GPT-2 style models with context lengths of up to 32k tokens, without any observed degradation in model quality [3].

3.2 Theme 2

With the development of large-scale language models

(LLMs), researchers are constantly exploring new ways to improve the performance and generative power of these models. A feasible study is that we can apply Sketching techniques to LLM models to improve efficiency [2]. However, Beurer-Kellner, L et al. proposed a novel hinting technique called Prompt Sketching [4]. This technique optimizes the generation process by allowing LLMs to predict the values of multiple variables in a template, providing better control and efficiency.

Prompt Sketching is a new prompting paradigm designed to address the limitations of traditional sequential prompting methods. In traditional approaches, models generate intermediate results and final answers through sequential queries, but these approaches often result in disconnected and lengthy intermediate responses [8]. In contrast, Prompt Sketching responds to prompts by predicting the values of multiple variables in a template, enabling users to provide a framework for reasoning through intermediate commands, leading to better overall results [4].

Specifically, Prompt Sketching involves several key steps: the first is template creation: the user creates a template containing multiple variables that will be predicted by the model during generation. The second is a variable prediction: the model not only completes the current prompt during the generation process but also predicts the values of multiple variables specified in the template. The third is intermediate instruction insertion: during the generation process, intermediate instructions provided by the user are inserted to guide the model's reasoning process.

The researchers introduced Prompt Sketching by developing two advanced sketch-aware decoders, VAR (Variable-Level Beam Search) and BEAMVAR (Variable-Grid Beam Search). These decoders, inspired by the study of Post and Vilar [9], enhance the inference process by optimizing the overall template likelihood. They effectively tackle the issues of duplication and deviation, which are prevalent in traditional decoding methods.

Luca Beurer-Kellner et al. conducted an extensive experimental evaluation of Prompt Sketching, validating its effectiveness in several reasoning benchmark tasks [4]. These tasks included logical reasoning, arithmetic reasoning, and general question-and-answer tasks. The experimental results showed that Prompt Sketching outperformed existing sequential prompting schemes, such as direct questioning or chain thinking, on seven reasoning benchmark tasks. In tasks such as logical reasoning and arithmetic reasoning, Prompt Sketching significantly improves the accuracy of the model ultimately improving the accuracy of the task by 10% [10]. Additionally, this technology can control the reasoning process: the sketch-aware decoding program makes the reasoning process more controllable by optimizing the overall likelihood of the template [4]. Overall, Prompt Sketching revolutionizes the way LLMs deal with complex reasoning tasks by providing a structured and controlled approach to text generation. Advances in template-guided reasoning and sketch-aware decoding demonstrate significant accuracy and coherence improvements, making sketch prompting an invaluable tool in the development and application of LLMs.

4. Comparison and Analysis

PolySketchFormer and Prompt Sketching demonstrate how sketching techniques can be leveraged to improve the efficiency and effectiveness of Large Language Models (LLMs).

By integrating polynomial kernels and sketching techniques, PolySketchFormer successfully optimizes Transformer models. Polynomial kernels help to reduce computational complexity. While a sketching technique called Polynomial Sketching can reduce the dimensionality of the data. By using high-order polynomial kernels and approximate feature maps, the computational complexity of the self-attention mechanism is reduced, thereby speeding up training without sacrificing model quality [3]. Moreover, another study called Prompt Sketching introduces a new sketching technique that can enhance LLMs efficiency. By predicting multiple variables within predefined templates, Prompt Sketching allows for more controlled and coherent text generation. This method not only improves accuracy in reasoning tasks but also provides users with greater influence over the model's decision-making process, leading to more insightful and contextually appropriate outputs [4].

PolySketchFormer and Prompt Sketching highlight how sketching techniques can be used in various occasions related to LLMs. Furthermore, PolySketchFormer shows how Sketching techniques can be combined with other techniques to maintain both the accuracy and efficiency of the training of LLMs. As for Prompt Sketching, it indicates the possibility of enhancing the reasoning ability of LLMs by using sketching techniques. These insightful studies can inspire us to study sketching techniques better.

Both PolySketchFormer and Prompt Sketching show how sketching techniques can be used as a means to improve the efficiency of LLMs. In the current AI training study field, the training of a model is always time-consuming, and making it more efficient has always been the focus of many researchers. We see how powerful sketching techniques can be when addressing this problem.

ISSN 2959-6157

5. Challenges and Future Directions

Looking ahead, the continued refinement and adoption of sketching techniques may help to further develop the current LLMs. Future research may explore deeper integrations of polynomial kernels and advanced sketching methods to tackle even larger datasets and more complex language tasks. Moreover, researchers may explore the possibility of combining Sketching techniques with even more other techniques to solve the possible problems brought by using Sketching techniques alone, like loss of accuracy. In the future, it is also possible to apply sketching techniques in other models in addition to LLMs, like Large Multimodal Models (LMMs).

However, many challenges remain. The actual application of sketching techniques can be very hard as decoding can be one of them [4]. Future research should focus on developing more robust sketching algorithms to minimize accuracy loss and explore their integration with other advanced techniques. Sketching techniques are also mainly applied to LLMs which leaves us with almost no experience of how to apply them on other models.

6. Conclusion

PolySketchFormer and Prompt Sketching represent important advances in leveraging sketching techniques to improve the performance of large language models. PolySketchFormer optimizes the computational efficiency of Transformer models, while Prompt Sketching improves the accuracy and control of text generation in reasoning tasks. This technology makes Sketching a valuable tool for the development and application of LLMs. As research in this field continues to develop, further improvements and integration of sketching methods are expected to reveal new possibilities to improve the efficiency and effectiveness of LLMs. Applying sketching techniques to other model architectures (such as LMMs) is also expected to extend these benefits to a wider range of AI applications. Ultimately, continued advances in sketching techniques will play an important role in shaping LLMs and their capabilities in natural language processing and other fields.

Acknowledgement
Junyin Zhang, Yuheng Shen, Zecen Ding, and Yitian Wan contributed equally to this work and should be considered co-first authors.

References

[1] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2024). *A Comprehensive Overview of Large Language Models* (arXiv:2307.06435). arXiv. http://arxiv.org/abs/2307.06435

[2] Woodruff, D. P. (2014). Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends® in Theoretical Computer Science*, *10*(1–2), 1–157. https://doi.org/10.1561/0400000060

[3] Kacham, P., Mirrokni, V., & Zhong, P. (2024). *PolySketchFormer: Fast Transformers via Sketching Polynomial Kernels* (arXiv:2310.01655). arXiv. http://arxiv.org/abs/2310.01655

[4] Beurer-Kellner, L., Fischer, M., & Vechev, M. (2023). Prompting Is Programming: A Query Language for Large Language Models. *Proceedings of the ACM on Programming Languages*, *7*(PLDI), 1946–1969. https://doi.org/10.1145/3591300

[5] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, *33*, 1877–1901. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[6] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling Laws for Neural Language Models* (arXiv:2001.08361). arXiv. http://arxiv.org/abs/2001.08361

[7] Charikar, M., Chen, K., & Farach-Colton, M. (n.d.). *Finding Frequent Items in Data Streams*.

[8] *Reynolds L., McDonell K. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm | Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. (n.d.). Retrieved July 13, 2024, from https://dl.acm.org/doi/10.1145/3411763.3451760#core-collateral-purchase-access

[9] Post, M., & Vilar, D. (2018). Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1314–1324). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1119

[10] Ling, W., Yogatama, D., Dyer, C., & Blunsom, P. (2017). Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems. In R. Barzilay & M.-Y. Kan (Eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 158–167). Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1015