Evaluating Tokenizers for Memory, Compute and Cache Efficiency: YouTokenToMe v/s TikToken

Divy Patel, Smit Shah, Sujay Chandra Shekara Sharma, Venkata Abhijeeth Balabhadruni
(dspatel6, spshah25, schandrashe5, balabhadruni)@wisc.edu
Group - 10

1 Introduction

Natural language processing (NLP) is an interdisciplinary subfield of computer science and linguistics. The primary goal of NLP is to give computers the ability to interpret, manipulate, and generate human language. The field of NLP enjoys a long history dating back to the famous paper by Alan Turing [12] which proposes the task of understanding and processing human text as a barometer of intelligence for a computer. However, the field has gained renewed focus and importance in recent years with the development of models based on neural networks that were able to outperform statistical methods that were considered to be state-of-the-art [1]. As a result, significant strides have been made in NLP, and NLP techniques are now used for a variety of problems like text classification and extraction, named entity recognition, sentiment analysis, and more recently, building AI chatbots [4].

One fundamental step in most NLP tasks is that of tokenization which involves breaking down raw text into smaller atomic units called tokens [15]. These tokens could be words, subwords, or even characters depending on the tokenizer being employed and the NLP task. Tokenization forms the foundation of many NLP tasks by transforming raw textual data into a structured format that can be easily processed by machine learning models. Since this forms one of the first steps in most NLP tasks, any errors in tokenization can lead to incorrect interpretations of the text and further errors in the downstream NLP tasks [3]. Given the importance of tokenization to the overall success of an NLP task and the large number of tokenizers available, it becomes vital to evaluate tokenizers on a variety of metrics and choose the tokenizer that is best suited to a specific task [5].

In this project, we explore the popular open-source tokenizer YouTokenToMe [14] and evaluate it using various profiling tools to get more insights about its performance bottlenecks. We also aim to identify potential areas for improvement by analyzing its memory utilization at each level of the caching hierarchy. Finally, we compare YouTokenToMe to OpenAI's TikToken [6] which acts as the benchmark for our experiments.

In the past, researchers have compared multilingual models with monolingual models by testing them on tasks designed for one language, using different tokenizers. They found that using monolingual tokenizers with multilingual models for these tasks can improve downstream performance of the pipeline. They use measures like F1 score and accuracy, to evaluate the performance of various downstream models in context with monolingual tokenizers utilized upstream.

In addition to that, there has been some work evaluating the compression factor of various tokenizers. They show how tokenizers with higher compression rates can increase the context lengths of the models and hence, can increase the amount of input text that can be ingested as part of inference. Also, they show how the compression rate of tokenizers affects the downstream memory usage and compute usage.

Current work focuses on evaluating the compression factor and accuracy, F1 score of tokenizers. None of the recent work evaluates the memory and cache efficiency of tokenizers. If we wish to verify the system level performance and identify the bottlenecks there is no existing work for the same. We plan to see the impact of two tokenizers (YouTokenToMe and Tiktoken) on system performance, specifically the following:

1. How does an increase in word count have an impact on CPU cache performance?
2. How does an increase in word count have an impact on the latency of the tokenizer?
3. How does CPU time change over different tokenization algorithms?
4. How does CPU utilization change over different tokenization algorithms?
5. How does cache utilization change over different tokenization algorithms?

6. How do the above changes vary on different languages like Hindi, Chinese, and Japanese?

We expect to see what function in the tokenizer algorithm is a bottleneck causing an increase in cache and CPU latency. We plan to visualize different steps in the algorithm that are causing a performance drop.

2 Related Work

2.1 Performance evaluation of Tokenizers

The papers in this subsection evaluate tokenizers either using various accuracy metrics or specific scenarios. [2] examines tokenizer accuracy through precision, recall, and F1 scores. [13] compares tokenizers to determine their effectiveness during the inference phase of machine learning models.

2.2 Evaluation of Tokenizers in case of multilingual models.

The papers in this subsection evaluate tokenizers in the context of multilingual models. [8] investigates how tokenizers affect multilingual language models compared to their monolingual counterparts. [10] examines a multilingual tokenizer employed in a large language model. [7] delves into the significance of tokenizers for multilingual language models.

2.3 Tokenizers Analysis on Specific Foreign Languages.

The papers in this subsection evaluate tokenizers' performance across different languages. [11] analyzes the performance of various tokenizers on the Turkish language. [9] assesses a specific tokenization technique for the Chinese language.

None of the aforementioned papers focuses on analyzing the performance and efficiency of tokenizers within the context of hardware utilization, particularly focusing on cache and memory traffic. This analysis aims to identify areas for improvement in tokenizer implementation, particularly in terms of better resource utilization.

3 Timeline and Evaluation Plan

For evaluating our project we plan to do the following:

- Perform an analysis of YouTokenToMe compared to the base model, TikToken
- Compute how much traffic is cached in L1, L2 caches and efficiency of compute

We plan to utilize the 4 weeks in the following way:

- 1st week: Set up bench-marking tools and environment for running Tiktoken and YouTokenToMe.
- 2nd week: Compute cached data percentage and cache utilization for Tiktoken and YouTokenToMe.
- 3rd week: Compare Tiktoken and YouTokenToMe on performance aspects and accuracy.
- 4th week: Compile the results, plot visualizations, and publish the report.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003.

[2] S. Choo and W. Kim. A study on the evaluation of tokenizer performance in natural language processing. *Applied Artificial Intelligence*, 37(1):2175112, 2023.

[3] T. Horsmann and T. Zesch. LTL-UDE @ EmpiriST 2015: Tokenization and PoS tagging of social media text. In P. Cook, S. Evert, R. Schäfer, and E. Stemle, editors, *Proceedings of the 10th Web as Corpus Workshop*, pages 120–126, Berlin, Aug. 2016. Association for Computational Linguistics.

[4] D. Khurana, A. Koli, K. Khatter, and S. Singh. Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, 82(3):3713–3744, Jan 2023.

[5] S. J. Mielke, Z. Alyafeai, E. Salesky, C. Raffel, M. Dey, M. Gallé, A. Raja, C. Si, W. Y. Lee, B. Sagot, and S. Tan. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *CoRR*, abs/2112.10508, 2021.

[6] OpenAI. Tiktoken. https://github.com/openai/tiktoken, 2023. Accessed: March 8, 2024.

[7] A. Petrov, E. L. Malfa, P. H. S. Torr, and A. Bibi. Language model tokenizers introduce unfairness between languages, 2023.

[8] P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, and I. Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models, 2021.

[9] C. Si, Z. Zhang, Y. Chen, F. Qi, X. Wang, Z. Liu, Y. Wang, Q. Liu, and M. Sun. Sub-Character Tokenization for Chinese Pretrained Language Models. *Transactions of the Association for Computational Linguistics*, 11:469–487, 05 2023.

[10] F. Stollenwerk. Training and evaluation of a multilingual tokenizer for gpt-sw3, 2023.

[11] C. Toraman, E. H. Yilmaz, F. Şahinuç, and O. Ozcelik. Impact of tokenization on language models: An analysis for turkish. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 22(4), mar 2023.

[12] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950.

[13] O. Uzan, C. W. Schmidt, C. Tanner, and Y. Pinter. Greed is all you need: An evaluation of tokenizer inference methods, 2024.

[14] VK.com. Youtokentome. https://github.com/vkcom/YouTokenToMe, 2019. Accessed: March 8, 2024.

[15] J. Webster and C. Kit. Tokenization as the initial phase in nlp. pages 1106–1110, 01 1992.