13th International Conference on Applied Informatics
Eszterházy Károly Catholic University
Eger, Hungary, February 19–21, 2026

huToken: A fast BPE tokenizer for language models

Roland Gunics[ab], **Máté Norbert Molnár**[ab], **Enikő Héja**[a], **Noémi Liget-Nagy**[a], **Ádám Kovács**[b], **Mátyás Osváth**[a]

[a]ELTE Research Centre for Linguistics
{gunics.roland,molnar.mate,heja.eniko,ligeti-nagy.noemi,osvath.matyas}@nytud.elte.hu

[b]Eszterházy Károly Catholic University
{gunics.roland,molnar.mate.norbert,kovacs2.adam}@uni-eszterhazy.hu

Abstract

Language models rely on tokenization - the process of segmenting text into smaller units and representing them as numerical sequences - to preprocess textual data. Consequently, the computational speed, memory usage and throughput of tokenization can impact the overall performance of language model training and inference. Among the various tokenization methods, Byte Pair Encoding (BPE) and its byte-level variant are widely used compression-based algorithms that ensure full vocabulary coverage without introducing unknown tokens or out-of-vocabulary issues [3]. Currently, there are three mainly used implementation for BPE tokenization: the HuggingFace's Tokenizers library[1], OpenAI's tiktoken[2] and Google's SentencePiece[3]. All three libraries are highly optimised, deterministic and reliable across platforms, providing Python bindings for ease of use. The first two are implemented in Rust, while SentencePiece is implemented in C++. Although these programming languages are very efficient, few studies systematically evaluate and compare their efficiency. Arguments can be made that further optimisations - including memory, speed and energy efficiency - are possible with C [2].

In this work, we present huToken[4], an open-source BPE tokenizer implemented in C and offering competitive performance in terms of speed compared to existing

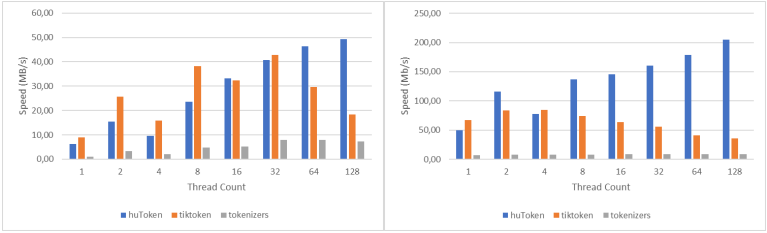
[1]https://github.com/huggingface/tokenizers
[2]https://github.com/openai/tiktoken
[3]https://github.com/google/sentencepiece
[4]https://github.com/matyasosvath/hutoken

implementations. It supports end-to-end tokenizer training and inference with Python bindings and the capability to load HuggingFace tokenizers.

huToken improves processing speed through a custom single-pass parser and the use of efficient string-matching algorithms, such as Aho–Corasick[1], in the core encoding and decoding pipelines. While naive implementations of the BPE merging algorithm require rescanning the whole text, we optimize this process using a priority queue and linked list structure.

We evaluate huToken against the two of the three aforementioned libraries using datasets spanning 1 MB to 1 GB and parallel configurations ranging from 1 to 128 threads on a dual-socket system equipped with two Intel(R) Xeon(R) Gold 5218 (2.30 GHz) processors, each one providing 16 physical cores and 32 hardware threads. Our experimental results show that huToken achieves performance comparable to tiktoken and delivers substantially higher encoding and decoding throughput than the tokenizers library, surpassing tiktoken at thread counts above 16 (see Fig. 1). Specifically, huToken achieves more than 30 MB/s encoding speed and almost 150 MB/s decoding speed on a 1 GB text corpus using 16 threads.

Overall, the initial results demonstrate that a C-based tokenizer implementation can match or exceed the performance of widely used Rust and C++ libraries, particularly under higher degrees of parallelism. These findings highlight the potential of huToken as an efficient and scalable alternative for high-throughput tokenization in large-scale language model training and inference workflows.



(a) Encoding speed **(b)** Decoding speed

Figure 1. Average encoding and decoding throughput measured over 10 runs on a 1 GB text corpus.

References

[1] A. V. Aho, M. J. Corasick: *Efficient string matching: an aid to bibliographic search*, Commun. ACM 18.6 (June 1975), pp. 333–340, ISSN: 0001-0782, DOI: 10.1145/360825.360855, URL: https://doi.org/10.1145/360825.360855.

[2] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, J. Saraiva: *Energy efficiency across programming languages: how do energy, time, and memory relate?*, in: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2017, Vancouver, BC, Canada: Association for Computing Machinery, 2017, pp. 256–267, ISBN: 9781450355254, DOI: 10.1145/3136014.3136031, URL: https://doi.org/10.1145/3136014.3136031.

[3] R. Sennrich, B. Haddow, A. Birch: *Neural Machine Translation of Rare Words with Subword Units*, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ed. by K. Erk, N. A. Smith, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725, doi: 10.18653/v1/P16-1162, url: https://aclanthology.org/P16-1162/.