dhruvin_IARCO_2025_Junior_International - Dhruvin Bhandari.pdf

by Mr Adnan

Submission date: 14-Oct-2025 02:32PM (UTC+0300)
Submission ID: 2780191743
File name: dhruvin_IARCO_2025_Junior_International_-_Dhruvin_Bhandari.pdf (110.7K)
Word count: 1533
Character count: 9403

Name – Dhruvin Bhandari

Email – bhandaridhruvin7@gmail.com

Institution Name – Divine Child International School

Country – India

Category – Junior

Class/Year – AS level /11th equivalent

Subjects – Physics, Chemistry, Maths, Further Maths, Biology, Computer Science, and English as a general language

Research Topic - Parallel Hierarchical Tokenisation: A Linear-Time Encoder for BPE-Compatible LLMs

Abstract

Tokenisation is the first step every large language model (LLM) must perform, yet state-of-practice algorithms such as Byte-Pair Encoding (BPE) still run in O(n log n) time and require the entire input before they can emit a single token. These constraints inflate end-to-end latency, hinder streaming applications, and waste compute on devices where power and memory are scarce. I propose the Parallel Hierarchical Tokeniser (PHT), an automata-based encoder that traverses a prefix trie once, achieving amortised O(n) throughput while preserving exact token boundaries and ID outputs for existing BPE vocabularies (assuming deterministic merge application and consistent vocabulary encoding). By allowing concurrent chunk processing and incremental updates, PHT offers a practical path to real-time inference without model retraining. This project will deliver the first systematic study of linear-time, streaming-safe tokenisation for modern LLMs and quantify its impact on typical deployments.

1. Introduction

Large-scale models such as GPT-3 and LLaMA rely on subword vocabularies to balance vocabulary size with representational power. Although compression-oriented designs have served NLP well, recent empirical analyses and theoretical work expose tokenisation as a hidden scalability bottleneck: worst-case runtimes reach $O(n^2)$ under adversarial input, and even optimised libraries remain strictly sequential. No published method to date guarantees linear-time performance while preserving identical token boundaries and token IDs under the same vocabulary, leaving a clear gap between infrastructure needs and available solutions. Addressing this gap would immediately benefit latency-sensitive services such as voice assistants, retrieval-augmented generation, and on-device inference.

2. Literature Review

Tokenisation algorithms have evolved from character-level and word-level approaches toward subword methods that balance vocabulary efficiency with semantic preservation. BPE, the dominant algorithm in modern LLMs, including GPT-3 and GPT-4, iteratively merges frequent character pairs but exhibits O(n log n) average-case complexity and degrades to $O(n^2)$ under adversarial patterns. WordPiece and Unigram Language Model tokenisation offer alternative merge strategies but maintain similar sequential processing requirements. Recent benchmarking reveals tokenisation consumes 2-15% of inference latency, with proportionally higher impact on short inputs and streaming applications. Approximation-based parallel approaches sacrifice exact

boundary reproduction, rendering them incompatible with existing trained vocabularies. No prior work achieves provably linear-time complexity while maintaining bit-exact compatibility with standard BPE, representing the research gap PHT addresses.

---

3. Research Question and Hypotheses

Central Question: Can tokenisation be redesigned to guarantee linear-time, streaming-compatible performance without altering downstream model behaviour?

Primary Hypothesis: PHT will achieve O(n) amortised time complexity while producing token sequences identical to standard BPE in both boundaries and IDs.

Secondary Hypotheses:

- PHT will demonstrate 2-4× throughput improvement over optimised sequential BPE
- Parallel execution will scale near-linearly with CPU cores (up to 8 cores)
- Streaming mode will sustain sub-2ms incremental latency
- Memory overhead will remain within 10% of baseline implementations

Objectives:

- Formalise PHT and prove equivalence to BPE on any given vocabulary
- Implement reference encoder with multithreading and incremental modes
- Benchmark throughput, memory, and correctness across diverse corpora
- Explore trade-offs between chunk size, overlap, and latency
- Release code and datasets under an open licence

---

4. Methodology

4.1 Algorithm Design

PHT constructs a deterministic prefix trie from existing vocabularies, then performs greedy maximal-munch parsing in a single left-to-right pass. For parallel execution, input is sliced into overlapping windows with locally resolved boundaries, assuming token merges do not exceed the bounded maximum length. A rolling state buffer enables streaming-source compatibility. Implementation will be developed in Rust (~1000 lines) with three modes: sequential (correctness baseline), parallel (multi-threaded), and streaming (incremental processing).

4.2 Data Collection

Corpus	Size	Purpose	Source
WikiText-103	108M tokens	Natural language baseline	
GitHub Code	20GB subset	Code tokenisation stress test	
Multilingual	German/Romanian, 25M tokens each	Cross-linguistic evaluation	
Synthetic Adversarial	10MB repetitive patterns	Worst-case analysis	Custom generation

Three BPE vocabularies (32K, 50K, 100K tokens) will be trained on domain-specific corpora using HuggingFace Tokenisers. Data will be partitioned 80/10/10 for vocabulary training, hyperparameter tuning, and held-out evaluation.

4.3 Performance Metrics and Analysis

Primary Metrics: Tokens-per-second throughput, wall-clock latency, peak memory (RSS), and token-level agreement with HuggingFace baseline (must achieve 100%).

Statistical Framework: Paired t-tests will compare mean throughput between PHT and baselines ($\alpha = 0.05$), with Shapiro-Wilk normality verification. Non-parametric Mann-Whitney U tests will be applied if normality assumptions fail. Two-way repeated-measures ANOVA will assess algorithm × thread count × dataset interactions, with Tukey HSD post-hoc tests and Bonferroni correction. Effect sizes (Cohen's d) and 95% confidence intervals will quantify practical significance.

Visualisation: Grouped bar charts (throughput by algorithm/dataset), line graphs (scalability vs. core count), box plots (streaming latency distributions), and heatmaps (chunk size × overlap ablations).

4.4 Baselines and Ablations

Results will be compared against HuggingFace v0.19 and GitHub's linear-BPE-Rust. Ablations will systematically vary overlap width (0-64 characters), thread count (1, 2, 4, 8, 16), input entropy (natural language, code, random), and vocabulary size (8K, 32K,

50K, 100K). Evaluation will be conducted on commodity 8-core Intel/AMD CPUs within a three-month timeline.

---

5. Data Analysis

Hypothesis Testing: Each secondary hypothesis will be evaluated quantitatively. H1 (2-4× speedup) is supported if 95% CI for the throughput ratio falls within [2.0, 4.0]; H2 (linear scaling) requires $R^2 > 0.90$ for throughput vs. core count regression; H3 (sub-2ms latency) demands 95th percentile < 2ms; H4 (<10% overhead) needs median memory ratio < 1.10 with $p < 0.05$.

Correctness Verification: Token-level agreement must reach 100% on all test data; any discrepancy triggers debugging. Property-based testing with randomised inputs will validate trie construction equivalence.

Failure Analysis: Incorrect tokenisations will be taxonomised by error type (encoding issues, boundary failures, vocabulary gaps) to identify algorithmic limitations versus implementation bugs.

---

6. Expected Outcomes and Impact

PHT is expected to halve or quarter tokenisation time on typical hardware, scale near-linearly with core count, and sustain sub-2ms incremental latency while emitting token streams identical to standard BPE in both boundaries and IDs. Such gains translate directly into lower serving costs, faster user interactions, and broader accessibility for edge devices. Because PHT requires neither vocabulary rebuilds nor model fine-tuning, adoption cost is minimal, positioning this work as an immediate improvement to the LLM tooling stack and a springboard for future hardware-aware parsers. The systematic benchmarking and open-source release will provide the NLP community with both theoretical insights and practical tools for improving inference efficiency, particularly benefiting real-time applications such as conversational AI, live transcription analysis, and on-device inference, where latency critically impacts user experience.

---

7. Conclusion

This research addresses a critical bottleneck in modern LLM deployment by proposing the first provably linear-time, streaming-compatible tokenisation algorithm that maintains exact compatibility with existing BPE vocabularies. The rigorous evaluation framework across diverse corpora, statistical validation, and open-source implementation will enable immediate adoption while establishing methodology for future preprocessing optimisations. Future work will explore SIMD acceleration,

morphologically complex languages, and integration with vocabulary-free architectures.

---

8. Potential Limitations

- Focus on CPU tokenization; no GPU or TPU evaluation
- Integration into complex production pipelines may require additional engineering
- Large vocabularies could increase memory consumption
- Speedups may be less significant for very short inputs where tokenization latency is minimal

---

References

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," arXiv:1706.03762, Jun. 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

T. B. Brown *et al.*, "Language models are few-shot learners," arXiv:2005.14165, May 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

H. Touvron *et al.*, "Llama: Open and efficient foundation language models," arXiv:2302.13971, Feb. 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

GitHub AI and ML Team, "So many tokens, so little time: Introducing a faster, more flexible byte-pair tokenizer," GitHub Blog, Jul. 2024. [Online]. Available: https://github.blog/ai-and-ml/llms/so-many-tokens-so-little-time-introducing-a-faster-more-flexible-byte-pair-tokenizer/. [Accessed: Jul. 4, 2025].

V. Zouhar, C. Meister, J. L. Gastaldi, L. Du, T. Vieira, M. Sachan, and R. Cotterell, "A formal perspective on byte-pair encoding," arXiv:2306.16837, Jun. 2024. [Online]. Available: https://arxiv.org/abs/2306.16837

T. Wolf *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," arXiv:1910.03771, Oct. 2020. [Online]. Available: https://arxiv.org/abs/1910.03771

S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," arXiv:1609.07843, Sep. 2016. [Online]. Available: https://arxiv.org/abs/1609.07843

D. Kocetkov *et al.*, "The stack: 3 tb of permissively licensed source code," arXiv:2211.15533, Nov. 2022. [Online]. Available: https://arxiv.org/abs/2211.15533

dhruvin_IARCO_2025_Junior_International - Dhruvin Bhandari.pdf

ORIGINALITY REPORT

11%	11%	8%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	pssg.cs.umd.edu Internet Source	2%
2	assets-eu.researchsquare.com Internet Source	2%
3	Xin Huang, Yu Fang, Mingming Lu, Yao Yao, Maozhen Li. "An Annotation Model on End-to-End Chest Radiology Reports", IEEE Access, 2019 Publication	2%
4	Christian Munley, Aaron Jarmusch, Sunita Chandrasekaran. "LLM4VV: Developing LLM-driven testsuite for compiler validation", Future Generation Computer Systems, 2024 Publication	1%
5	Submitted to University College London Student Paper	1%
6	Submitted to Universitas Sultan Ageng Tirtayasa Student Paper	1%
7	Submitted to University of Birmingham Student Paper	

		1 %
8	repository.tudelft.nl Internet Source	1 %
9	arxiv.org Internet Source	1 %
10	export.arxiv.org Internet Source	1 %
11	www.frontiersin.org Internet Source	1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	Off		