Where Open Tokenizers Break Tamil: Pre-tokenizer Cuts, Missing Merges and Mid-Character Tokens

Draft, not for circulation

Jayakumar Gopalan
Zhagaram Labs
jay@zhagaramlabs.com

[TODO: submission date]

Abstract

Tamil is written in an abugida: most letters are a consonant followed by a combining vowel sign or *pulli* (virama), stored by Unicode as separate code points. Recent work showed that pre-tokenizer patterns matching letters with `\p{L}+` split such scripts at every combining mark. We measure what this and two related failures cost Tamil in 14 widely used open tokenizers, on 3,000 held-out passages of web and literary Tamil. Our metrics, released as track D4 of the Uraikal benchmark, separate three kinds of damage: boundaries forced by the pre-tokenizer, tokens that end inside a UTF-8 character, and learned boundaries inside a letter, which Tamil morphology sometimes justifies. Qwen3's pre-tokenizer cuts 98.8% of Tamil words and the tokenizer needs 9.99 tokens per word, 4.1× Gemma 3 and 6.0× a Tamil-trained tokenizer. Its successor Qwen3.5 uses a mark-aware pattern and a larger vocabulary and needs 4.28, still 1.75× Gemma 3. A mark-aware pattern is not enough on its own: DeepSeek-V3 has no pre-tokenizer cuts, yet 21.8% of Tamil words contain a token that ends mid-character. The harness is open source.

1 Introduction

A tokenizer decides how much text a language model sees per step, how much each request costs, and which units the model must learn to compose. Tokenization is known to be uneven across languages [1, 13, 18], and Tamil, spoken by more than 80 million people, is among the languages that pay most.

Part of the reason is script. The word தமிழ் (*tamil*, "Tamil") is three letters to a reader, த, மி and ழ், but five Unicode code points and 15 UTF-8 bytes, because the vowel sign ி and the pulli ் are combining marks (category `M`). A pre-tokenizer that matches words as runs of `\p{L}` stops at every mark. This was reported for OpenAI's cl100k pattern in 2024 [2], shown for Tamil, Sinhala and Hindi pre-tokenizers [21], and recently characterised as a fertility ceiling across 26 languages [17] and as a limit on vocabulary extension [6].

We do not claim the mechanism. We ask what it, and two related failures, cost Tamil in the tokenizers people use today. Our contributions:

1. **D4**, a Tamil tokenizer track whose metrics tell three failures apart: pre-tokenizer cuts, mid-character tokens, and learned boundaries inside a letter (§3).

2. **Measurements of 14 open tokenizers**, including current models (Qwen3, Qwen3.5, DeepSeek-V3, gpt-oss, Gemma 3, Sarvam), on held-out web and literary Tamil (§4).

3. **Evidence that a mark-aware pattern is necessary but not sufficient**: the Qwen3 to Qwen3.5 change more than halves Tamil token counts, while a mark-aware tokenizer can still split a fifth of Tamil words mid-character (§5).

2 Background and related work

Tamil in Unicode. Tamil has 12 vowels, 18 consonants, 216 consonant–vowel letters (உயிர்மெய்) and the ஃ. Unicode encodes the vowels and consonants as letters (`Lo`) and the 11 vowel signs, the pulli and the length mark ◌ௗ as combining marks (`Mc`/`Mn`), so a consonant–vowel letter is two code points.

Pre-tokenization. Byte-level BPE [15, 19] first splits text with a regular expression and learns merges only within the pieces, so a boundary the pre-tokenizer creates can never be merged away. The choice of pattern measurably changes compression and downstream quality [7]. Velayuthan and Sarveswaran [21] showed that GPT-2, GPT-4 and Llama 3 pre-tokenizers split Tamil words at the pulli and proposed grapheme-based BPE. Regmi et al. [17] compared `\p{L}+` with `[\p{L}\p{M}]+` across 26 languages, derived a fertility floor that needs no training, noted that o200k already uses the mark-aware class, and found the letters-only class in most popular text-generation models. Chaudhary [6] showed that adding merges cannot go below that floor for Nepali.

Indian-language tokenizer evaluations. Tamil fertility has been reported in model and tokenizer papers: 9.75 for Qwen in PARAM-1 [14], close to our 9.99 for Qwen3, and 2.50 for Gemma 3 in MUTANT [16], close to our 2.44. Other evaluations cover the 22 official Indian languages with one sample each [20], train new Indic tokenizers [5], or constrain BPE by morphology [4]. Tamil-Llama extended LLaMA's vocabulary with Tamil tokens [3]; tokenizer transfer methods initialise the new embeddings [8, 12].

Mid-character tokens. Byte-level vocabularies contain tokens that are partial UTF-8 sequences [11], and such vocabularies can always generate ill-formed UTF-8 [9]. We measure how often they occur inside Tamil words.

What is new here. A Tamil-specific measurement on held-out text of the tokenizers in current use, with the three failures separated; the Qwen3 to Qwen3.5 comparison; and the rate of mid-character tokens in a mark-aware tokenizer. [TODO: check Moon et al. (ICML 2026) on UTF-8 validity and VerChol (arXiv 2603.05883) before submission.]

3 The D4 track

Data. **D4.web** is 2,000 passages (113,875 words) drawn from shard `data-52` of the verified Tamil split of Sangraha [10] at revision `8b813c3f` (CC BY 4.0), a shard we hold out of all our own tokenizer training. **D4.literature** is 1,000 passages (48,044 words) from Project Madurai, a volunteer archive of classical and modern Tamil literature; we publish its scores but not its text. Passages are lines of at least 20 words, cleaned (Unicode NFC, invisible characters removed, legacy-order vowel signs repaired) and kept only if at least 90% of their letters are Tamil.

Tokenizer	Vocab	Tokens/word web	Tokens/word lit.	Bytes/ token	Tamil words cut (%) letter	Tamil words cut (%) pre-tok.	Tamil words cut (%) byte
Agaram-2.5B	64k	1.68	1.93	14.7	20.3	0	<0.1
IndicBERT v2†	250k	1.76	2.08	14.0	11.8	<0.1	0
MuRIL	197k	1.89	2.21	13.1	9.1	<0.1	0
BLOOM	251k	2.15	2.41	11.5	41.7	0	<0.1
Gemma 3	262k	2.44	2.69	10.1	25.4	0	<0.1
Sarvam-30B	262k	2.44	2.69	10.1	25.4	0	<0.1
Sarvam-1	68k	2.46	2.77	10.0	26.2	0	0.3
XLM-R	250k	2.48	2.79	9.9	16.2	0	4.6
mT5	250k	2.58	2.72	9.6	27.8	0	13.3
gpt-oss (o200k)	200k	3.19	3.41	7.7	65.0	0	2.2
Mistral Nemo	131k	3.77	3.89	6.5	77.4	0	5.9
Qwen3.5	248k	4.28	4.34	5.8	64.5	0	3.2
DeepSeek-V3	129k	4.96	4.94	5.0	87.9	<0.1	21.8
Qwen3	152k	9.99	10.02	2.5	98.8	98.8	99.5

Table 1: D4 results, sorted by tokens per word on D4.web. Bytes per token and the cut rates are for D4.web; D4.literature follows the same pattern. No tokenizer produced more than 0.2 unknown tokens per 1,000. †IndicBERT v2 fails the round trip only because its decoder keeps WordPiece continuation markers.

Metrics. For each tokenizer and slice:

- **Tokens per word** over whitespace-separated words, and **UTF-8 bytes per token**, read next to the vocabulary size.
- **Pre-tokenizer cut**: the share of Tamil words in which the pre-tokenizer, run alone, places a boundary immediately before a vowel sign, pulli or length mark. Always a defect.
- **Byte cut**: the share of Tamil words containing a token that ends inside a UTF-8 character, detected as consecutive tokens whose character offsets overlap. Always a defect.
- **Letter cut**: the share of Tamil words with any token boundary immediately before a mark. A diagnostic, not a defect: Tamil suffixes often begin inside a letter (மரத்தை = மரத்த் + ஐ, "the tree" + accusative), so a learned cut there can follow the grammar.
- **Unknown tokens** per 1,000 tokens, and **round trip**: the share of passages whose decoding equals the input, ignoring whitespace.

Tokenizers. We load 14 tokenizers from the Hugging Face Hub with `trust_remote_code=False`. Llama 3.1 and Aya Expanse are gated and were not included in this draft [TODO: add both].

4 Results

Table 1 and Figure 1 show a sixfold spread. The tokenizers trained with Tamil in mind lead: Agaram (1.68 tokens per word with only 64k entries) and IndicBERT v2 (1.76). Large multilingual vocabularies (Gemma 3, BLOOM, XLM-R, mT5) need 2.2–2.6. The regex-based byte-level tokenizers trail, and Qwen3 is an outlier at 9.99 tokens per word and 2.5 bytes per token, less than one Tamil character per token on average. Every tokenizer needs more tokens on literary than on web text, except DeepSeek-V3, which is equally poor on both.

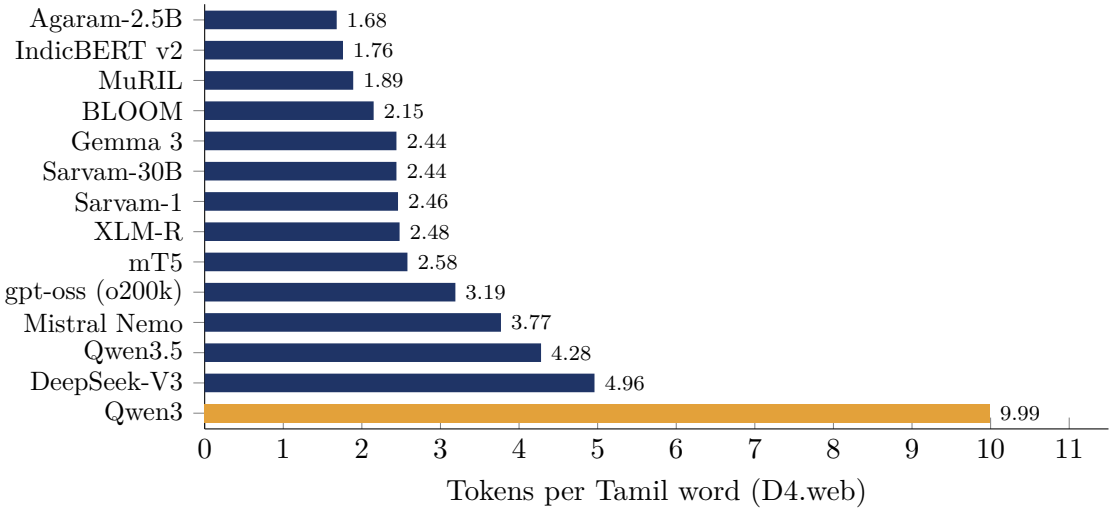


Figure 1: Tokens per word on D4.web. Fewer is better at a given vocabulary size. Qwen3, highlighted, is the only tokenizer with more than 0.1% pre-tokenizer cuts.

5 Analysis

The pattern accounts for Qwen3. Qwen3 matches words with

```
[^\r\n\p{L}\p{N}]?\p{L}+
```

so a vowel sign ends the run and attaches to the next piece: தமிழ் becomes தம | ிழ | ். This happens to 98.8% of Tamil words, and 99.5% contain a byte cut. The gpt-oss (o200k), Mistral Nemo, DeepSeek-V3 and Qwen3.5 patterns all include `\p{M}` in their letter classes and make almost none (<0.1%), consistent with [17].

Qwen3 to Qwen3.5. Qwen3.5 switched to `[\p{L}\p{M}]+` and grew its vocabulary from 152k to 248k entries. Tamil tokens per word fell from 9.99 to 4.28 and byte cuts from 99.5% to 3.2%. Both changes landed together, so this comparison cannot say how much each contributed (§6). Qwen3.5 still needs 1.75× the tokens of Gemma 3.

A mark-aware pattern is not sufficient. DeepSeek-V3 has no pre-tokenizer cuts yet splits 21.8% of Tamil words mid-character: its byte-level vocabulary contains too few Tamil merges to reach character boundaries. mT5 (13.3%) and XLM-R (4.6%) show the same effect through byte fallback. The pattern decides which merges are *possible*; the vocabulary decides which were *learned*.

Letter cuts are not the target. Even tokenizers that need fewer than 2.5 tokens per word place a boundary inside a letter in 9–42% of Tamil words, many where a suffix begins (மரத்தை). We therefore treat pre-tokenizer and byte cuts as defects and letter cuts as a diagnostic.

Shared vocabularies. Sarvam-30B and Gemma 3 produce identical tokens per word to two decimal places on both slices; 239,316 of Sarvam-30B's 262,144 entries also appear in Gemma 3's vocabulary.

Cost. At equal price per token, a Tamil request costs about 4.1× more on Qwen3 than on Gemma 3, and a context window holds about a quarter as much Tamil.

6 Separating pattern from vocabulary

[TODO: Train byte-level BPE with the Qwen3 pattern and with `[\p{L}\p{M}]+` at 2–3 fixed vocabulary sizes, on Sangraha shards other than `data-52`, and report all D4 metrics. This isolates the pattern's effect that the Qwen3 to Qwen3.5 comparison confounds with vocabulary growth. CPU only.]

7 Limitations

Tokens per word is a proxy; the real test is the quality of models trained with each tokenizer, which needs GPU time [TODO: bits-per-byte follow-up]. Words are whitespace-separated. D4.literature cannot be released, so its numbers can be checked only by us. Two widely used tokenizers are missing because they are gated. We measured the smallest Qwen3.5 checkpoint's tokenizer, which we assume the family shares [TODO: confirm].

8 Release

The metrics, cleaning code and data preparation are in `uraikal` (Apache 2.0; `https://github.com/zhagaramlabs/uraikal`, `pip install uraikal`); the command `uraikal tokenizers` reproduces Table 1. [TODO: D4.web on Hugging Face.]

References

[1] Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9904–9923, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.614. URL `https://aclanthology.org/2023.emnlp-main.614/`.

[2] ajaykg. Combining marks and indic vowel marks within words are being split breaking all indic languages and most languages except english and cjks. GitHub issue #292, openai/tiktoken, 2024. URL `https://github.com/openai/tiktoken/issues/292`. Opened 5 May 2024.

[3] Abhinand Balachandran. Tamil-llama: A new tamil language model based on llama 2, 2023. URL `https://arxiv.org/abs/2311.05845`.

[4] Maharaj Brahma, N J Karthika, Atul Singh, Devaraj Adiga, Smruti Bhate, Ganesh Ramakrishnan, Rohit Saluja, and Maunendra Sankar Desarkar. MorphTok: Morphologically grounded tokenization for indian languages, 2025. URL `https://arxiv.org/abs/2504.10335`.

[5] Maharaj Brahma, N J Karthika, Rajat Verma, Nagasai Saketh Naidu, Rohit Saluja, Maunendra Sankar Desarkar, and Ganesh Ramakrishnan. Multilingual tokenization through the lens of indian languages: Challenges and insights, 2025. URL `https://arxiv.org/abs/2506.17789`.

[6] Aarjan Chaudhary. Merge-based vocabulary extension stalls at the pre-tokenizer: An english-preserving repair for nepali, tested in continued pretraining, 2026. URL https://arxiv.org/abs/2610.33016.

[7] Gautier Dagan, Gabriel Synnaeve, and Baptiste Roziere. Getting the most out of your tokenizer for pre-training and domain adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 9784–9805, 2024. URL https://proceedings.mlr.press/v235/dagan24a.html.

[8] Konstantin Dobler and Gerard de Melo. FOCUS: Effective embedding initialization for monolingual specialization of multilingual models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13440–13454, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.829. URL https://aclanthology.org/2023.emnlp-main.829/.

[9] Preston Firestone, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. UTF-8 plumbing: Byte-level tokenizers unavoidably enable LLMs to generate ill-formed UTF-8. In *Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=8ExXncFpf6.

[10] Mohammed Safi Ur Rahman Khan, Priyam Mehta, Ananth Sankar, Umashankar Kumaravelan, Sumanth Doddapaneni, Suriyaprasaad B, Varun G, Sparsh Jain, Anoop Kunchukuttan, Pratyush Kumar, Raj Dabre, and Mitesh M. Khapra. IndicLLMSuite: A blueprint for creating pre-training and fine-tuning datasets for Indian languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15831–15879, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.843. URL https://aclanthology.org/2024.acl-long.843/.

[11] Sander Land and Max Bartolo. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11631–11646, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.649. URL https://aclanthology.org/2024.emnlp-main.649/.

[12] Benjamin Minixhofer, Edoardo M. Ponti, and Ivan Vulić. Zero-shot tokenizer transfer. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/532ce4fcf853023c4cf2ac38cbc5d002-Abstract-Conference.html.

[13] Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/74bb24dca8334adce292883b4b651eda-Abstract-Conference.html.

[14] Kundeshwar Pundalik, Piyush Sawarkar, Nihar Sahoo, Abhishek Shinde, Prateek Chanda, Vedant Goswami, Ajay Nagpal, Atul Singh, Viraj Thakur, Vijay Dewane, Aamod Thakur, Bhargav Patel, Smita Gautam, Bhagwan Panditi, Shyam Pawar, Madhav Kotcha, Suraj Racha, Saral Sureka, Pankaj Singh, Rishi Bal, Rohit Saluja, and Ganesh Ramakrishnan. PARAM-1 BharatGen 2.9b model, 2025. URL https://arxiv.org/abs/2507.13390.

[15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI,

2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[16] Souvik Rana, Arul Menezes, Ashish Kulkarni, Chandra Khatri, and Shubham Agarwal. MUTANT: A recipe for multilingual tokenizer design, 2025. URL https://arxiv.org/abs/2511.03237.

[17] Sajal Regmi, Siddhartha Pudasaini, and Chetan Phakami Pun. Vowel signs are not letters: A pre-tokenization ceiling on multilingual tokenizer fertility, 2026. URL https://arxiv.org/abs/2608.26449.

[18] Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.243. URL https://aclanthology.org/2021.acl-long.243/.

[19] Rico Sennrich, Barry Haddow, and Alexandra 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)*, pages 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162/.

[20] S. Tamang and D. J. Bora. Evaluating tokenizer performance of large language models across official indian languages, 2024. URL https://arxiv.org/abs/2411.12240.

[21] Menan Velayuthan and Kengatharaiyer Sarveswaran. Egalitarian language representation in language models: It all begins with tokenizers. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5987–5996, Abu Dhabi, UAE, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.400/.