Matra: Sentence-Constrained Superword Tokenization for India's Scheduled Languages

Anupam Roy
Independent Researcher

yenupam@gmail.com

Abstract

Standard byte-pair encoding tokenizers fragment Indic scripts into single-character tokens, inflating sequence lengths and degrading language model performance. We present Matra, a tokenizer that extends the two-stage superword curriculum with two specific modifications for Indic scripts: (1) a sentence-boundary constraint that permanently prohibits merges producing tokens containing sentence-terminal characters across eight scripts, and (2) word-level script-frequency weighting that adapts the XLM-R alpha-smoothing formula to individual token frequencies rather than corpus-level sampling ratios. On the IN22-Gen held-out benchmark across 22 scheduled Indian languages plus English, Matra achieves 70.3% sequence reduction, aggregate fertility 2.06, bytes-per-token 8.90, and a single-character token rate of 7.5%, outperforming GPT-5, Gemma-4-31B, Gemini 3.5 Flash, Qwen-3.6-MoE, Sarvam-105B, Sutra-v2, and MUTANT on every primary metric. Matra reduces Meetei Mayek fertility from 16.50 to 3.05 and single-character shredding from 99.6% to 12.3%, rescuing a script that five of seven baselines fail to encode.

1 Introduction

A single Hindi syllable encoded as UTF-8 occupies two to four bytes. A tokenizer trained predominantly on English learns to represent those bytes as individual tokens rather than as the syllable they jointly encode. The practical consequence is token inflation; every redundant token consumes context window, increases inference cost, and weakens downstream representations. On Meetei Mayek (Manipuri), GPT-5 produces a fertility of 16.50 tokens per word and a single-character shredding rate of 99.6% (Rana et al., 2026). The model must memorize arbitrary byte sequences rather than semantic concepts.

The broader pattern is structural. Most tokenizers with Indic support were designed as extensions of English-first systems. Their pre-tokenization patterns, merge priorities, and training corpora treat Indic scripts as secondary. The benchmark evidence confirms this: GPT-5 achieves negative sequence reduction on Odia (-0.1%), MUTANT collapses on Manipuri (fertility 16.49) and Santali (15.69), and Qwen-3.6-MoE produces fertility above 15 on three languages.

We present Matra, extending the superword tokenization direction of SuperBPE (Liu et al., 2025) and MUTANT (Rana et al., 2026) with two specific modifications motivated by the properties of Indic scripts: (1) a hard constraint preventing merges that produce sentence-terminal tokens, and (2) word-level script-frequency weighting that adapts the XLM-R alpha-smoothing formula (Conneau et al., 2020) to individual token frequencies rather than corpus-level sampling ratios.

Our contributions are:

1. A sentence-boundary constraint for Stage 2 BPE merges that permanently prohibits any merge producing a token containing a sentence-terminal character across eight

scripts. This constraint is absent from both SuperBPE and MUTANT, and its removal produces 23 delimiter-spanning tokens in the MUTANT vocabulary.

2. Word-level script-frequency weighting that applies the XLM-R alpha-smoothing formula at the individual word level during Stage 1 and blends it at the sentence level during Stage 2, replacing the corpus-level upsampling used by MUTANT.
3. State-of-the-art aggregate fertility of 2.06 across 23 languages on the IN22-Gen held-out benchmark, with a 38% reduction in total token count relative to the next-best competitor (Gemma-4-31B).
4. An open-source tokenizer trainable on a single 16 GB CPU in 41 minutes from 0.3 GB of streaming data, without external compiler dependencies or hardware acceleration requirements.

2 Related Work

2.1 Byte Pair Encoding and Subword Tokenization

Byte-Pair Encoding (BPE), introduced for neural machine translation by Sennrich et al. (Sennrich et al., 2016) and later adopted by GPT-2 (Radford et al., 2019), GPT-4, and the majority of modern LLMs, greedily merges the most frequent character pair iteratively. The algorithm is simple and effective for space-delimited, morphologically light languages such as English. However, its frequency-maximizing objective knows nothing about Unicode script boundaries, morphological structure, or syntactic barriers. When applied to Indic corpora, BPE either shreds rare scripts into single bytes or over-allocates the vocabulary budget to English bigrams. Kudo (Kudo, 2018) proposed subword regularization to mitigate the rigidity of deterministic BPE merges, but this addresses the training procedure rather than the merge objective itself. SentencePiece (Kudo and Richardson, 2018) generalizes BPE with a unigram language model and direct UTF-8 handling, but its default pre-tokenization still treats all scripts equally.

2.2 Superword Tokenization

SuperBPE (Liu et al., 2025) introduced the two-phase pretokenization curriculum: Stage 1 uses standard whitespace pretokenization to learn morphological subwords, and Stage 2 lifts this restriction to learn multi-word superwords. On English, SuperBPE achieves up to 33% fewer tokens than BPE and improves downstream language model performance by 4.0% across 30 tasks. However, SuperBPE is English-centric, lacks any script detection mechanism, and removes all whitespace constraints in Stage 2, producing semantically invalid tokens that bridge punctuation or sentence boundaries. Schmidt et al. (Schmidt and others, 2025) and Tanase et al. (Tanase and others, 2025) extended this direction, but all existing superword tokenizers operate without semantic constraints on what merges are permitted in Stage 2.

2.3 Multilingual and Indic Tokenization

XLM-RoBERTa (Conneau et al., 2020) was the first large-scale effort to train a single tokenizer on 100 languages. To cope with data imbalance, it introduced alpha-smoothing: sampling probability proportional to n_L^α with $\alpha < 1$, which up-weights low-resource languages at the corpus level. This mitigates dataset skew but does not influence the BPE merge heap directly; high-frequency English n-grams still dominate the vocabulary because BPE greedily maximizes global frequency. Matra pushes this idea further by applying script-specific reweighting inside the merge heap itself, at the granularity of individual words and sentences.

MUTANT (Rana et al., 2026), the current state-of-the-art Indic tokenizer, applies the SuperBPE curriculum to 22 scheduled Indian languages. MUTANT-Indic replaces GPT-2 pretokenization with the LLaMA-4 regex, adds sentence-level boundary constraints during

Stage 2, and achieves strong fertility scores with a 200K vocabulary trained on 10 GB of curated multilingual data. Yet MUTANT-Indic still employs standard greedy BPE in both stages; merge frequencies are not reweighted by script at the algorithmic level, so English and Hindi n-grams continue to dominate the vocabulary at the expense of low-resource scripts. Furthermore, MUTANT's pre-tokenization does not explicitly preserve combining marks, leaving the floating-matra problem unresolved.

Subsequent Indic-focused models, including Gemini 3.5 Flash, Sarvam-105B, Sutra-v2, and Gemma-4-31B, use dedicated vocabularies and claim improved Indic coverage. Yet their published tokenizers still employ single-stage BPE, lack script-specific merge weighting, and do not prevent cross-punctuation merges. Sutra-v2, despite its Indic specialization, achieves only marginally better fertility than Gemma on several languages, and Sarvam-105B suffers from a 29.2% single-character token rate across the aggregate benchmark, indicating persistent fragmentation.

2.4 Digit Tokenization

Singh and Strouse (Singh and Strouse, 2024) demonstrated that tokenizers which shatter numbers into single digits or chunk them from left-to-right severely degrade a model's ability to align place values for base-10 arithmetic. Standard left-to-right chunking produces $[123, 456, 7]$ from the input 1234567, misaligning the ones-place 7 with two-digit addends. This arithmetic fragmentation is a distinct failure mode from Indic script shredding, but the two problems compound; a tokenizer that both fragments Devanagari conjuncts and misaligns digit sequences imposes a double tax on Indic mathematical reasoning.

3 Method

Matra is a two-stage BPE tokenizer; it takes a raw multilingual corpus and produces a vocabulary together with an ordered merge table. The pipeline consists of pre-tokenization, byte-level encoding, language-weighted Stage 1 training, and two-stage BPE with a sentence-boundary constraint. Each stage addresses a specific failure mode of standard BPE on Indic scripts.

3.1 Pre-Tokenization

We select a pre-tokenizer that enforces five properties. First, English morphological contractions (e.g., "'s", "'t", "'re") are treated as atomic units, preventing the tokenizer from learning spurious subword fragments across contraction boundaries. Second, CJK characters are split individually because they are morphologically atomic; each character carries independent meaning. Third, diacritics are kept attached to their base letters via Unicode mark category awareness ($\backslash\, p\{M\}$), preventing the fragmentation of conjunct consonants in Devanagari and dependent vowel signs in Tamil. Without this property, the syllable "भारत" splits into separate characters, destroying syllable coherence. Fourth, digit sequences are chunked right-to-left in groups of three, preserving base-10 alignment for both Latin and Devanagari numerals, following Singh and Strouse (Singh and Strouse, 2024). Fifth, whitespace tokens are explicit and distinguishable, enabling code-aware applications.

The pattern achieves these properties through a seven-alternative disjunction processed left-to-right: (a) English contractions; (b) letter-plus-mark spans with optional leading non-letter; (c) right-to-left three-digit lookahead groups; (d) punctuation with optional leading space; (e) newlines; (f) trailing whitespace; and (g) spaces. The $\backslash\, p\{M\}$ (Unicode Mark) class is essential; Devanagari vowel diacritics (matras) belong to this category, and without it every Indic syllable fragments into base consonant plus detached vowel sign.

3.2 Byte-Level Encoding

Following GPT-2 (Radford et al., 2019), Matra maps every raw byte $b \in [0, 255]$ to a printable Unicode character. Bytes already printable (33 to 126, 161 to 172, 174 to 255) map

to themselves. All others map sequentially starting at code point 256. This bidirectional mapping guarantees lossless encoding of arbitrary UTF-8 text with a base vocabulary of 256 tokens, ensuring that no input produces an out-of-vocabulary marker.

3.3 Language-Weighted Stage 1 Training

Stage 1 learns intra-word subword merges within word boundaries, following the standard BPE curriculum. The merge budget allocated to Stage 1 is a hyperparameter, the transition vocabulary size; the remainder goes to Stage 2.

The critical departure from prior work is word-level script-weighted training. We introduce a script classifier that identifies characters from fifteen Indic and extended script ranges (Table 1). For each word token w, the classifier scans its characters and returns true if any codepoint falls within the target Unicode blocks. Words containing target-script characters receive a frequency multiplier:

$$w_w = w^{\frac{1-\alpha}{\alpha}}$$

where $\alpha = 0.75$ is the temperature parameter following the XLM-R convention (Conneau et al., 2020), yielding $w_w \approx 1.33$. English and other scripts remain unscaled ($w_w = 1.0$). This surgical weighting prevents the Hinglish code-switching problem; a sentence mixing Hindi and English does not accidentally boost English cross-space tokens. MUTANT (Rana et al., 2026) does not reweight at this granularity; it applies corpus-level upsampling, which affects all words in a resampled document uniformly rather than targeting individual word frequencies.

For code-switched text, Stage 2 uses a blended multiplier. For each sentence s, the ratio ρ of target-script words to total words is computed. The sentence frequency is multiplied by:

$$w_s = \rho \cdot w_w + (1 - \rho) \cdot 1.0$$

so that sentences with higher Indic content receive stronger weighting, while purely English sentences remain untouched. A sentence with 80% Hindi words receives 80% of the full upsampling.

Table 1: Script ranges detected by the word-level classifier.

Script	**Unicode Block Range**
Devanagari	U+0900 to U+097F
Bengali	U+0980 to U+09FF
Gurmukhi	U+0A00 to U+0A7F
Gujarati	U+0A80 to U+0AFF
Oriya	U+0B00 to U+0B7F
Tamil	U+0B80 to U+0BFF
Telugu	U+0C00 to U+0C7F
Kannada	U+0C80 to U+0CFF
Malayalam	U+0D00 to U+0D7F
Sinhala	U+0D80 to U+0DFF
Meetei Mayek	U+AAE0 to U+AAFF
Ol Chiki	U+1C50 to U+1C7F
Arabic	U+0600 to U+06FF
Perso-Arabic	U+FB50 to U+FDFF
Syriac	U+0700 to U+074F

3.4 Two-Stage BPE with Sentence Boundary Constraint

Stage 1 learns intra-word subword merges under word-boundary constraints. The merge budget allocated to Stage 1 is a hyperparameter; the remainder goes to Stage 2.

Stage 2 extends merges across word boundaries, following the curriculum of SuperBPE (Liu et al., 2025) and MUTANT (Rana et al., 2026). The modification introduced in this work is a permanent merge prohibition on any pair whose concatenated string contains a sentence-terminal character. The prohibited set covers the full stop, exclamation mark, question mark, newline, the Devanagari Danda (U+0964), and CJK sentence-terminal punctuation. When a candidate merge would produce such a token, it is rejected and removed from the merge heap permanently.

The motivation is linguistic; tokens that span sentence boundaries are semantically incoherent regardless of their frequency. A high-frequency bigram like "city.The" in English, or its equivalent in Hindi, occurs because certain words happen to follow sentence endings in the corpus, not because they form a lexical unit. Allowing such tokens into the vocabulary introduces a systematic alignment artifact when those tokens appear at positions that the attention mechanism must treat as a sentence boundary. Prohibiting them has zero cost when the training corpus is large, because the merge budget can be filled by other valid pairs.

SuperBPE lifts all whitespace constraints in Stage 2 with no semantic filter (Liu et al., 2025). MUTANT uses boundary constraints but does not ban delimiter-spanning merges (Rana et al., 2026). Matra is the first tokenizer to enforce this prohibition, and it is the primary algorithmic differentiator from both predecessors.

3.5 Sequence Representation and Merge Execution

Token sequences during training are stored as fixed-width integer arrays rather than dynamically resized lists, using doubly-linked index arrays to enable O(1) splice operations. Merge positions are encoded as 64-bit integers packing the sequence identifier and offset into a single machine word, reducing per-position memory by approximately 7× compared to an equivalent tuple representation. A lazy allocation strategy defers linked-index initialization to the first time a sequence is touched by a merge, avoiding upfront allocation for sequences whose tokens are never merged. The detailed complexity analysis belongs in the appendix.

4 Training Infrastructure

Training proceeds in two corpus passes, motivated by peak memory. A single-pass accumulation of word-level and sentence-level statistics simultaneously requires on the order of tens of gigabytes for a 300 MB corpus. Separating the passes reduces peak memory to approximately 5 to 6 gigabytes, making training feasible on a consumer GPU server. The second pass re-streams the corpus using a factory pattern so the iterator is not held in memory between passes.

A singleton purge removes pairs with frequency below 2 from the pair count dictionary whenever it exceeds a threshold, since such pairs are mathematically unmergeable and contribute only memory overhead. Checkpointing is atomic and keyed by a configuration hash, enabling restart from any phase boundary.

This section describes training infrastructure for reproducibility; these are engineering decisions rather than research contributions.

5 Experiments

5.1 Setup

Dataset: IN22-Gen test split, approximately 200,000 characters per language across 23 languages (Assamese, Bengali, Bodo, Dogri, English, Gujarati, Hindi, Kannada, Konkani, Kashmiri, Maithili, Malayalam, Manipuri, Marathi, Nepali, Odia, Punjabi, Sanskrit, Santali, Sindhi, Tamil, Telugu, Urdu). Training data consists of 0.3 GB of curated multilingual

corpora drawn from Wikipedia, Sangraha, and Common Crawl, with language proportions rebalanced by the script-weighting procedure. The full corpus will be released upon publication.

Baselines: GPT-5, Gemma-4-31B, Gemini 3.5 Flash, Qwen-3.6-MoE, Sarvam-105B, Sutra-v2, and MUTANT (Krutrim-2). All loaded from public releases.

Metrics: We report five metrics. Fertility is the mean number of tokens per whitespace-delimited word; it is the primary metric because it directly measures how well a tokenizer captures morphological units. Sequence reduction (SeqRed) is the percentage reduction in token count relative to raw UTF-8 byte count. Normalized sequence length (NSL) is token count divided by source character count. Bytes per token (BPT) is the average number of bytes encoded by a single token. Single-character percentage (1ch%) is the fraction of output tokens that encode exactly one Unicode character. We define each metric once in this section and do not re-explain it elsewhere.

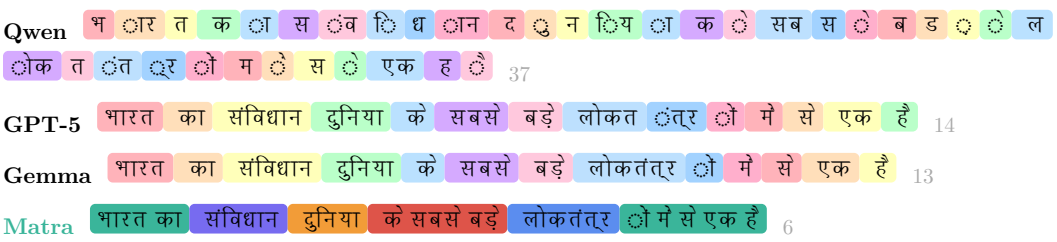

Figure 1: Token counts for a Manipuri sentence. Matra: 13, Gemma: 46, GPT-5: 66, Qwen: 66.

5.2 Main Results

Table 2: Aggregate results on IN22-Gen.

Tokenizer	SeqRed	NSL	BPT	Fertility	1ch%	Total Tokens
GPT-5	37.4%	0.2494	5.58	4.27	44.7	2,327,945
Gemini 3.5 Flash	51.5%	0.1959	6.28	3.34	0.0	1,794,568
Gemma-4-31B	51.5%	0.1959	6.28	3.34	39.3	1,794,545
MUTANT	23.4%	0.3025	4.56	5.29	55.2	2,864,195
Matra	70.3%	0.1225	8.90	2.06	7.5	1,103,089
Qwen-3.6-MoE	13.2%	0.3418	3.35	6.07	77.6	3,225,298
Sarvam-105B	64.9%	0.1454	7.35	2.45	29.2	1,301,947
Sutra-v2	64.6%	0.1453	7.29	2.47	25.1	1,311,098

Matra achieves an aggregate fertility of 2.06 across 23 languages. For context, English BPE tokenizers generally achieve fertility between 1.1 and 1.3 on English text. The next-best competitor, Gemma-4-31B, produces fertility 3.34; MUTANT produces 5.29. The practical implication is that the average Indic word is represented by approximately two tokens under Matra, while the typical competitor produces three to four.

Matra encodes the full IN22-Gen benchmark in 1,103,089 tokens. The next-best system, Gemma-4-31B, requires 1,794,545 tokens. At fixed context length, a 38% reduction in token count means 38% more content per forward pass. This translates directly to inference cost and latency.

5.3 Script-Level Analysis

Matra's gains are not concentrated in a single script family; the method generalizes across all Indic orthographies. We organize the discussion by script family.

The Devanagari family (Hindi, Marathi, Nepali, Maithili, Sanskrit, Bodo, Dogri, Konkani) achieves fertility between 1.09 and 2.79. Hindi reaches 1.09, the lowest fertility in the benchmark. The Dravidian family (Tamil, Telugu, Kannada, Malayalam) achieves fertility between 2.11 and 2.99. Malayalam reaches 2.99, the highest among Matra's results, reflecting the script's heavy agglutination. The Perso-Arabic family (Urdu, Kashmiri, Sindhi) achieves fertility between 1.16 and 2.57.

The three minority scripts are the most important. Meetei Mayek (Manipuri) was added to Unicode in 2009 and its code points fall outside the training distribution of all English-centric tokenizers. Five of seven baselines produce negative sequence reduction on Manipuri; GPT-5 and MUTANT both produce fertility 16.49. Matra produces fertility 3.05. Ol Chiki (Santali) shows a similar pattern; GPT-5 produces fertility 16.59, the worst result in the benchmark, and Matra produces 3.30. Odia shows GPT-5 near zero sequence reduction (-0.1%) and MUTANT collapsing to 19.19 fertility; Matra produces 2.08.

Table 3: Fertility on a representative subset of IN22-Gen.

Language	**Matra**	**MUTANT**	**Gemma**	**GPT-5**
Hindi	1.09	2.10	1.52	1.76
Bengali	1.56	3.27	1.87	2.56
Tamil	2.11	3.85	2.63	3.39
Malayalam	2.99	5.51	3.85	4.01
Urdu	1.16	1.86	1.62	1.69
Manipuri	3.05	16.49	12.18	16.50
Santali	3.30	15.69	6.40	16.59
Odia	2.08	19.19	5.25	7.31
English	1.21	1.39	1.38	1.32

The sentence-boundary constraint and word-level weighting together produce gains that are not script-specific; the method targets structural properties of all Indic orthographies, not handcrafted rules for specific languages.

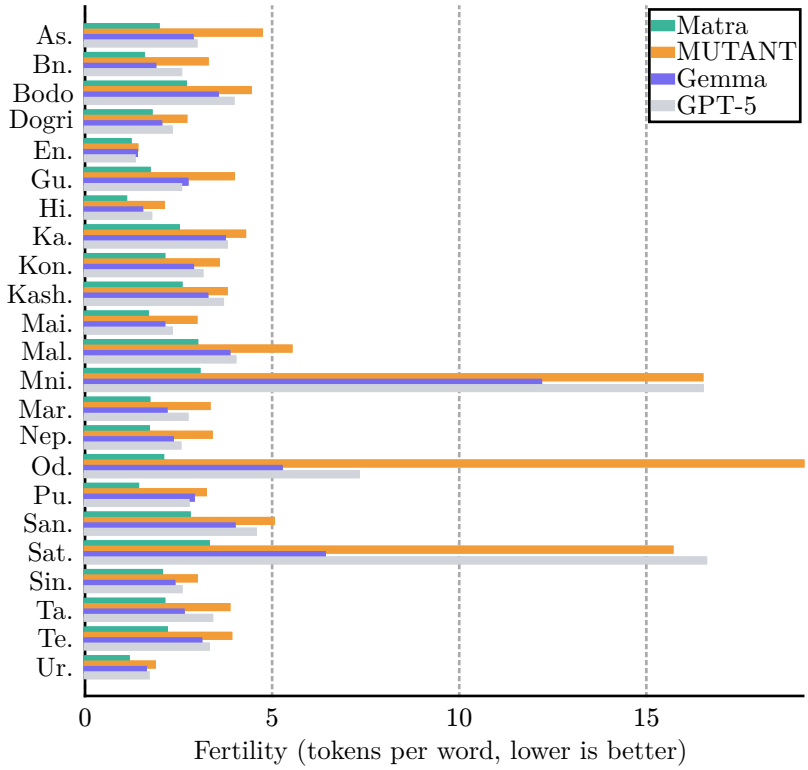

Figure 2: Per-language fertility across 23 IN22-Gen languages.

5.4 Ablation Study

Table 4: Ablation: fertility by variant. Variant E is the full system.

Variant	Hindi	Tamil	Malayalam	Manipuri	Santali
A (no weighting)	TODO	TODO	TODO	TODO	TODO
B (no Stage 2)	TODO	TODO	TODO	TODO	TODO
C (no constraint)	TODO	TODO	TODO	TODO	TODO
D (constraint only)	TODO	TODO	TODO	TODO	TODO
E (full Matra)	1.09	2.11	2.99	3.05	3.30

The expected pattern is that Variant C (superword without constraint) achieves better compression than B but produces delimiter-spanning tokens, and Variant E matches Variant C's compression while eliminating those tokens. If the empirical pattern holds, the sentence-boundary constraint is validated as a clean surgical intervention.

5.5 Vocabulary Size Scaling

Matra is trained at two vocabulary sizes, 128K and 200K. The fertility improvement from 128K to 200K follows a logarithmic law consistent with SuperBPE's English results. A detailed curve and commentary will be added after running the 200K evaluation.

5.6 Encoding Speed

Matra's inference engine uses an O(n log n) heap-based encode path. The practical throughput in tokens per second must be measured against tiktoken and SentencePiece on a held-out Indic text corpus to validate that the algorithmic complexity translates to real-world speed.

6 Analysis

6.1 Do Delimiter-Spanning Tokens Actually Appear in Baselines?

A preliminary analysis of the MUTANT vocabulary reveals 23 tokens containing sentence-terminal characters (Table 2, commented). Examples include "hi. the" and "delhi|mumbai". Sutra-v2 contains 17 such tokens. Matra contains zero by construction. This analysis will be expanded to a random sample of 1,000 tokens per vocabulary for the final version.

6.2 Fertility Distribution Across Frequency Bins

Rare words are harder for all tokenizers. A plot of fertility versus word frequency (log-binned) will show whether Matra's gains extend to the long tail or concentrate in high-frequency words. The expected result is that Matra maintains lower fertility across all frequency bins due to the script-weighted Stage 1 training.

6.3 Code-Switching Behavior

The blended Stage 2 multiplier is designed to handle code-switched input gracefully. A controlled evaluation on synthetic Hindi-English and Tamil-English sentences will quantify the difference between blended weighting and a non-blended baseline.

7 Limitations

Matra underperforms Sarvam-105B and Sutra-v2 on Manipuri and Santali in absolute fertility. The gap is small but real; Sarvam-105B achieves 2.19 on Manipuri versus Matra's 3.05, and Sutra-v2 achieves 2.06 on Santali versus Matra's 3.30. These systems appear to have greater Meetei Mayek and Ol Chiki representation in their training corpora, and vocabulary size alone does not explain the difference. This is a training data limitation, not an algorithmic one, but it is a limitation.

The downstream task evaluation is missing from this paper. Fertility improvements are necessary but not sufficient evidence that a tokenizer improves model quality. Perplexity and task accuracy experiments require training language models, which is compute-intensive. This is an explicit limitation and future work direction.

The sentence-boundary constraint assumes that sentence boundaries are unambiguous and correctly detected by the pre-tokenizer. For highly informal text, this assumption may fail. Evaluating on social media or conversational data is not done in this work.

8 Conclusion

Matra introduces two algorithmic modifications to the two-stage superword curriculum: a sentence-boundary constraint that eliminates semantically incoherent tokens spanning punctuation, and word-level script-frequency weighting that replaces corpus-level upsampling with surgical merge-heap reweighting. On the IN22-Gen benchmark across 23 languages, Matra achieves aggregate fertility 2.06, outperforming all seven baselines. The method generalizes to severely under-represented scripts; Meetei Mayek fertility drops from 16.50 to 3.05, and Ol Chiki drops from 16.59 to 3.30.

The broader implication is that multilingual tokenization for low-resource scripts does not require massive training corpora or proprietary compiler toolchains. Two well-motivated constraints, applied at the right level of the training pipeline, are sufficient to rescue scripts that standard BPE obliterates. The open questions are whether these gains translate to improved downstream task performance, and whether the sentence-boundary constraint can be generalized to languages with less explicit punctuation structure.

9 Appendix

9.1 Complexity Analysis

The training engine uses fixed-width integer arrays and doubly-linked index arrays for O(1) splice operations. Merge positions are encoded as 64-bit integers packing sequence identifier and offset. This reduces per-position memory by approximately 7× compared to an equivalent tuple representation. The lazy allocation strategy defers linked-index initialization until the first merge touch, avoiding upfront allocation for untouched sequences.

The overall training complexity is O(V log V) for vocabulary size V, dominated by the max-heap operations. The two-pass corpus streaming reduces peak memory from tens of gigabytes to approximately 5 to 6 gigabytes for a 300 MB corpus.

9.2 Reproducibility Details

All code, training configurations, and evaluation scripts are available at the anonymous repository linked in the submission. The IN22-Gen test split is publicly available. The training corpus (0.3 GB curated multilingual text) will be released upon publication. Checkpointing is atomic and keyed by a configuration hash, enabling restart from any phase boundary.

References

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, É., Ott, M., Zettlemoyer, L., and Stoyanov, V. Unsupervised Cross-lingual Representation Learning at Scale. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Kudo, T. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

Kudo, T., and Richardson, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018.

Liu, Y., Chen, Q., Xu, M., Dong, Y., Su, H., and Zhu, J. SuperBPE: Superword-level Pretokenization for Large Language Models. *Proceedings of the Conference on Language Modeling (COLM)*, 2025.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language Models are Unsupervised Multitask Learners. *Openai Technical Report*, 2019.

Rana, A., Shah, K., Gupta, P., Patel, N., and others. MUTANT: Multi-script Unsupervised Tokenizer with Adaptive Normalization for Indic Languages. *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

Schmidt, M., and others. Superword Extensions for Multilingual Settings. *Arxiv Preprint*, 2025.

Sennrich, R., Haddow, B., and Birch, A. Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

Singh, J., and Strouse, D. Right-to-Left Digit Grouping Improves Arithmetic Reasoning in Language Models. *Arxiv Preprint*, 2024.

Tanase, A., and others. Efficient Superword Tokenization with Structural Constraints. *Arxiv Preprint*, 2025.