Universal Byte-Level Encoding: UTF-8/UTF-16 Routing to Reduce Cross-Script Token-Budget Disparities

Hyunsik Kim
Samsung Research
hyunsik777.kim@samsung.com

Youngmoon Jung
Samsung Research
youngm.jung@samsung.com

Abstract

Byte-level byte-pair encoding (BBPE) tokenizers are attractive for multilingual large language models because they cover all Unicode text. In UTF-8-based BBPE, however, many scripts start from a higher fallback cost than English: when no learned merges apply, a multibyte character requires multiple byte-derived symbols. We call this worst-case pre-merge cost the *encoding floor*. A higher floor can increase token counts and per-request cost while shrinking usable context. Changing the whole text encoding can reduce this gap but make already-efficient English spans more expensive in mixed-script text. Thus, we propose Universal Byte-Level Encoding (UBE), a dual-alphabet tokenizer that keeps 1–2-byte UTF-8 characters in UTF-8 and routes 3–4-byte characters through UTF-16. This lowers the encoding floor for 3-byte Basic Multilingual Plane characters, used by many scripts that need more tokens than English, without raising it for any character. UBE changes only the symbols presented to byte-pair encoding, and a Unicode 17 audit confirms exact round trips for every scalar value and every input in three official test suites. UBE also composes with other boundary policies and morphology-based encodings. Across intrinsic evaluations, UBE lowers dispersion in English-normalized token-count ratios (Gini 6–14% lower on FLORES-200 at 16K–64K), reducing cross-lingual token-budget disparity. In multilingual language models, UBE matches BBPE's quality. In the main multilingual settings, UBE reduces token counts most where they are highest and also lowers English counts, yielding more usable context and faster prompt processing in content-matched benchmarks. Byte-level models benefit directly: UBE shortens their inputs by 29–33% in all 39 of the FLORES-200 languages written mostly in 3-byte characters (1.4–1.5 times as much text per context window), lowers byte-level Gini by 28%, and lengthens no input. Code is available at https://github.com/Samsung/UBE.

1 Introduction

Tokenization is often treated as a preprocessing detail, but in multilingual language modeling it also determines how fixed context and compute budgets are spent across languages. When a tokenizer fragments one language into many more units than another, semantically comparable content consumes more of the usable context window and requires processing longer sequences.

For autoregressive transformers, prompt token count determines the sequence length processed during prefill, so cross-lingual differences in tokenized length, which we call token-budget disparity under a fixed tokenizer, directly affect usable context, latency, and prompt-processing cost. This disparity is not merely a side effect of multilingual data imbalance; it is also a property of tokenizer design itself [Petrov et al., 2023, Arnett et al., 2025].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

We target the part of this disparity that the byte encoding alone imposes: under a shared tokenizer, semantically comparable text should not need systematically more tokens only because its script takes more bytes per character. In pay-per-token large language model (LLM) interfaces, the same gap also raises the price of a request [Ahia et al., 2023, Artola Velasco et al., 2026].

Petrov et al. [2023] show that, under widely deployed tokenizers such as GPT-4's cl100k_base, parallel Shan text can require up to $15\times$ as many tokens as English; by our calculation, a fixed 128K-token context then holds only as much Shan content as about 8.5K English tokens. The problem is especially acute for compact or on-device language models (LMs), where English-centric compact models often use 32K–50K tokenizers [Allal et al., 2025, Liu et al., 2024]; prior work also suggests that tokenizer quality matters more at smaller model scales [Goldman et al., 2024]. In FLORES-200 [NLLB Team, 2024], seven languages are written in the Tibetan, Myanmar, Ol Chiki, and Tifinagh scripts, which use 3-byte Basic Multilingual Plane (BMP) characters and have little training data; we call these seven *low-coverage BMP* (LC-BMP) languages. Under a compact 16K vocabulary trained on our 101-language corpus, a byte-level byte-pair encoding (BBPE) tokenizer averages 308.522 tokens per parallel sentence for the LC-BMP languages but only 34.863 for English (Appendix Table D.3), so the same context window holds only about one-ninth as much parallel content in these languages as in English. Tokenizer design therefore becomes especially consequential for compact multilingual LMs serving mixed-script users under fixed context budgets.

BBPE is a useful setting for isolating this issue because it is simple, reversible, and language-agnostic: every Unicode string can be represented using a fixed 256-symbol byte alphabet, and byte-pair encoding (BPE) merges can be learned without language-specific segmentation rules. This simplicity, however, comes with an asymmetry inherited from UTF-8: before any merges are learned, characters from different scripts are represented by byte sequences of different lengths. In BBPE, each character is first decomposed into UTF-8 bytes, and BPE merges then combine frequent adjacent byte-derived symbols into larger tokens. Because a finite multilingual vocabulary cannot cover all subword patterns that arise across languages and text domains, low-coverage spans continue to fall back toward byte-level decomposition. We define the *encoding floor* for this substrate-imposed fallback as a scalar-level fallback cost: the number of base byte symbols required for one Unicode scalar value under the underlying byte encoding when no learned merge covers it. It is not a lower bound on post-BPE token counts; learned merges can compress frequent characters and multi-character spans below the floor, while underrepresented spans often remain close to it. Under UTF-8, this floor is script- and code-point-dependent (Section 2, Table 1).

Recent work shows that replacing the byte-encoding substrate can lower this floor for some scripts [Kim et al., 2026]; however, a uniform replacement can also raise costs for spans already efficient under UTF-8 in mixed-script text. In this work, we propose Universal Byte-Level Encoding (UBE), a dual-alphabet BBPE tokenizer with deterministic per-character routing based on UTF-8 byte length. Characters encoded in 1–2 UTF-8 bytes stay on the UTF-8 path, while 3-byte BMP characters are routed through UTF-16, where they require 2 bytes. Because routing is per character rather than per language, UBE lowers the encoding floor for scripts most exposed to byte-level fragmentation without imposing the uniform 2-byte floor on already-efficient spans in mixed-script text. The method leaves BPE training and merge learning unchanged, preserves exact round-trip decoding for encoded text, and requires no language identification or model-architecture changes. Byte-level models benefit directly: UBE shortens their inputs by 29–33% in every FLORES-200 language written mostly in 3-byte characters, lengthens no input, and lowers byte-level Gini by 28% (Section 3.3).

Table 1: Encoding-floor values before BPE merges. BBPE16 [Kim et al., 2026] is a UTF-16-only baseline; bold marks UBE's target trade-off: ASCII stays at the 1-byte UTF-8 floor, while 3-byte BMP scripts (e.g., CJK, Devanagari, and Thai) fall from 3 to 2 via the UTF-16 Private Use Area (PUA) path; 4-byte non-BMP code points remain at a 4-symbol floor. CJK stands for Chinese, Japanese, and Korean.

	ASCII	Cyrillic/Arabic/Hebrew	CJK, Devanagari, Thai	Non-BMP
BBPE	1	2	3	4
BBPE16	2	2	2	4
UBE	**1**	2	**2**	4

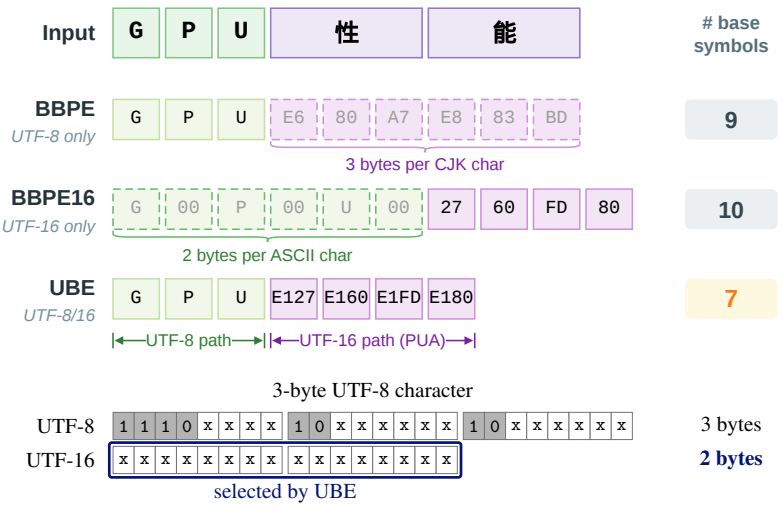

Figure 1: Pre-merge base-symbol decomposition of "GPU性能". UBE, BBPE, and BBPE16 use 7, 9, and 10 base symbols, respectively. Appendix A gives the full algorithm. Bottom: a 3-byte BMP code point under UTF-8 and UTF-16; gray cells are UTF-8 length and continuation bits (all length classes: Appendix Figure A.1).

2 Background and method

Encoding floor under standard BBPE. BBPE [Wang et al., 2020] converts text to UTF-8 bytes and learns BPE merges over the resulting byte stream, so each byte is one base symbol and BBPE's encoding floor is the UTF-8 byte length: 1 for ASCII, 2 for many Cyrillic, Arabic, and Hebrew characters, 3 for Chinese, Japanese, and Korean (CJK) and other 3-byte scripts, and 4 for non-BMP code points (Table 1). GPT-2-style implementations, following the encoder released with GPT-2 [Radford et al., 2019], first map each byte to a reversible visible symbol.

Design target and compatibility constraints. UBE targets one specific source of multilingual tokenization disparity in BBPE under finite vocabularies: 3-byte BMP scripts that remain close to byte fallback pay more per character than 1- and 2-byte scripts, and UTF-16 lowers their encoding floor from 3 to 2 bytes (Table 1). By design, UTF-8 is less compact for these scripts (Figure 1, bottom): a 3-byte sequence (`1110xxxx 10xxxxxx 10xxxxxx`) spends 8 of its 24 bits on length and continuation markers that make UTF-8 self-synchronizing, leaving 16 code-point bits, which UTF-16 stores in 2 bytes [Kim et al., 2026]. Routed UTF-16 bytes synchronize at the alphabet switch instead: the decoder pairs each run's bytes into 2-byte code units from its first symbol (Appendix A.4). Non-BMP 4-byte code points, including many emoji, are routed through the same deterministic branch but remain 4 base symbols as UTF-16 surrogate pairs; UBE therefore matches BBPE's 4-symbol encoding floor for these characters, and unless otherwise noted we focus on 3-byte BMP scripts. To preserve compatibility with existing BPE pipelines, we impose three constraints: exact reversibility, unchanged BPE training and vocabulary learning, and no explicit marker tokens for encoding choice, which would add base symbols and could increase token counts.

UBE: universal byte-level encoding. BBPE16 [Kim et al., 2026] demonstrated that replacing UTF-8 with UTF-16 lowers the CJK floor to 2 while preserving a language-agnostic byte-level representation, which is well matched to multilingual automatic speech recognition (ASR) and other settings dominated by CJK text. In the mixed-script LM setting studied here, however, the same global substrate also raises the ASCII floor to 2 (Table 1; Figure 1 illustrates the contrast on a mixed-script example). UBE avoids this trade-off with a single deterministic UTF-8/UTF-16 routing rule based on the number of bytes in each character's UTF-8 encoding:

$$r(c) = \begin{cases} \text{UTF-8} & \text{if } |\text{utf8}(c)| \leq 2, \\ \text{UTF-16} & \text{otherwise.} \end{cases}$$

Thus, the routing rule forms a complete partition of Unicode scalar values. The UTF-16 branch serializes code units as UTF-16LE without a byte-order mark. Here, "character" means one Unicode scalar value rather than a user-perceived grapheme cluster; combining-mark sequences and zero-width joiner sequences are routed scalar by scalar, with no additional normalization. Two disjoint 256-symbol alphabets—the GPT-2 byte alphabet for UTF-8 and a Private Use Area (PUA) block (U+E100–E1FF) for UTF-16[1]—let the decoder infer the active encoding from each symbol's alphabet membership, without marker tokens. The full encoder/decoder details, including pseudocode (Algorithms 1 and 2) and dual-buffer flush-on-switch decoding, are in Appendix A.

BPE compatibility and exact decoding. UBE changes the initial symbol stream presented to a standard BPE trainer: the base alphabet becomes the 512-symbol union of both byte alphabets, and the vocabulary is learned from that routed stream. The pair-counting interface and merge-selection criterion remain the standard BPE ones; UBE introduces no script-aware merge rule or marker token. UBE therefore wraps an unchanged BPE core with a routing pre-tokenizer, disjoint base alphabet, and custom dual-buffer tokenizer runtime. Proposition 1 in Appendix A.4 proves exact round-trip decoding. Empirically, UBE has no decoding failures on FLORES-200 across 204 languages and recovers every tested raw input in an exhaustive Unicode 17 audit spanning all Unicode scalar values and the official normalization, grapheme-break, and emoji test suites. Among the audited tokenizers, only UBE recovers every input: MYTE [Limisiewicz et al., 2024] recovers 1,138,306 of the 1,155,724 inputs, and SCRIPT-BPE [Land and Arnett, 2025] applies Unicode Normalization Form C (NFC) and filters Private-Use scalars, unassigned scalars, and Unicode 17 additions (Appendix Table E.7).

Alphabet size and marker-free design. Keeping ASCII at a 1-byte floor while mapping common 3-byte BMP UTF-8 characters to a 2-byte path requires the decoder to know which byte encoding produced each base symbol; the 512-symbol disjoint union alphabet provides this without markers, even under unrestricted BPE merges. A merge token may combine symbols from both alphabets (e.g., a Latin–CJK pair within one regex chunk); after the token is expanded, each base symbol's alphabet still identifies its encoding, so the dual-buffer decoder flushes at the alphabet boundary inside the token (Appendix D.5.3 reports how often such merges occur at each vocabulary size).

Vocabulary overhead. The trade-off is vocabulary budget: with total vocabulary size V and $S = 5$ shared special tokens, UBE leaves $(V - S - 512)$ merge slots instead of $(V - S - 256)$ for BBPE—a 256-slot difference at every V, an overhead of roughly $256/V$, still <1% at typical LM vocabulary sizes ($V \geq 32K$).

3 Experiments

3.1 Intrinsic tokenizer evaluation

Metrics and evaluation scope. Our primary intrinsic metric is the language-level token premium computed from sentence-level token counts on aligned text. For a tokenizer τ, language ℓ, and aligned sentence set S_ℓ, let $m_\tau(\ell) = \frac{1}{|S_\ell|} \sum_{s \in S_\ell} |\tau(s)|$ denote the mean number of tokens per sentence. We define the premium as $\pi_\tau(\ell) = m_\tau(\ell)/m_\tau(\text{en})$; thus $\pi_\tau(\ell) > 1$ means that aligned content occupies more positions in the context window than English under the same tokenizer. For dispersion statistics, the English reference row is excluded, so Gini and variance are computed over $\{\pi_\tau(\ell) : \ell \neq \text{en}\}$ (definitions in Appendix E.1; robustness corpora in Appendix D.5.4). These tokenizer-level statistics quantify allocation under fixed context and compute budgets: for semantically matched inputs, they determine how much context capacity and prefill computation each language consumes.

Tokenizer setup and evaluation. In the core UBE–BBPE comparison, both variants use the same BPE pipeline; only the initial symbol stream differs. We compare three fixed mC4 [Xue et al., 2021] raw-text shard configurations: `En-only` (encoding-stress), `6-lang` ({en, zh, ko, ja, hi, th}, narrow-coverage), and `101-lang` (full-coverage, used for the main matched LM comparison). The core matrix trains all three configurations at 16K/32K/64K vocabularies, and the `101-lang` vocabulary-scaling sweep extends to 128K and 256K; within each configuration–vocabulary pair, the raw input

[1] Using this PUA range for internal byte symbols does not introduce ambiguity: input characters in the same range are first encoded as UTF-16LE bytes and then mapped to those symbols (Appendix A.3).

data are held fixed across tokenizer families, so token-count differences reflect the tokenizer alone. The primary intrinsic benchmark is the FLORES-200 devtest split (1,012 parallel sentences $\times$ 204 language–script pairs, hereafter languages); pipeline details (shard-local counting, singleton filtering, merge aggregation), corpus construction, and per-configuration token-count statistics are in Appendix Sections B.1 and B.3.

FLORES-200 results. UBE reduces the Gini coefficient of token premiums in every setting of the `En-only`, `6-lang`, and `101-lang` $\times$ 16K/32K/64K core matrix, and also in the `101-lang` 128K and 256K extensions. Across the nine core comparisons, relative reductions range from 6% to 14% (Appendix Table D.1). The relative Gini reduction is largest at `En-only/16K` and smallest at `6-lang/64K`. Appendix Tables D.1 and D.3 give the full configuration and vocabulary sweeps. The three-way matrix identifies vocabulary size as the dominant source of variation: the reduction shrinks as the vocabulary grows in every configuration, and `6-lang` narrows it relative to `En-only`. Even at `101-lang/16K`, however, UBE still lowers Gini (-12.1%), showing that merge-budget dilution across many scripts keeps many languages near the encoding floor. A paired language bootstrap over non-English language-level premiums directly supports the dispersion claim: the 95% confidence interval (CI) for ΔGini (UBE $-$ BBPE) is below zero in all nine core comparisons, including [-0.0358, -0.0047] in the primary `101-lang/32K` setting (Appendix Table E.1). As a separate paired token-count check, two-sided Wilcoxon signed-rank tests over language-level mean token counts are significant in 8/9 core comparisons at $p < 0.01$. The same 8/9 token-count decision holds after Holm correction at $\alpha = 0.05$ and Benjamini–Hochberg false discovery rate (FDR) correction at $q = 0.05$ over the nine prespecified core comparisons. The only non-significant token-count comparison is `En-only/64K`, where UBE and BBPE have nearly matched language-level token counts ($p = 0.32$).

Encoding-floor mechanism under vocabulary scaling. Under `101-lang/32K`, UBE reduces Gini from 0.2183 to 0.1979 (-9.4%) with a mean UBE/BBPE ratio of 0.9920. The largest token-count reductions occur in low-coverage 3-byte scripts such as Santali (Ol Chiki), Central Atlas Tamazight (Tifinagh), and Tibetan, where UBE reduces the byte-level fallback cost from 3 bytes to 2 bytes per character (Figure 2b). Across all 204 languages, UBE produces fewer tokens in 189 languages and more tokens in 15 languages. Figure 2a shows that UBE's Gini stays below BBPE's from 16K to 256K, although the gap narrows at larger vocabularies (-1.9% at 256K), whereas under emulated commercial mixtures UBE still lowers Gini by 5.5–10.0% at 128K–200K (Appendix E.8); Appendix Table D.3 reports the full numerical sweep. The overall upward trend in Gini reflects heterogeneous compression gains among non-English languages (Appendix Sections D.2.1 and D.2.2). The half-data ablation (Appendix D.4) preserves per-language sampling proportions and thus tests corpus size rather than language mixture. Its full- and half-data curves nearly overlap across all four tokenizer families, so insufficient tokenizer-training volume alone does not explain the upward Gini trend. Section 3.3 instead varies the language mixture at a fixed 30 GiB. Appendix D.5 provides the per-language distribution view, and Appendix D.5.2 provides the script-family view of UBE's token-count changes.

Cross-dataset robustness and contamination. To test robustness on additional evaluation sets, we repeat the `101-lang/32K` comparison on three additional corpora (Table 2). The NLTK Unicode UDHR corpus (`udhr2`; 387 entries, many with no mC4 training data) shows a larger Gini reduction under UBE (-11.1%) than on FLORES-200 (-9.4%), consistent with the encoding-floor mechanism being most visible where many entries remain near the byte-level fallback floor. Because UDHR is widely reproduced and our mC4-overlap audit finds frequent UDHR prefix matches, we treat it as a public-text stress test rather than a clean held-out set (Appendix C.3). SIB-200 [Adelani et al., 2024], a topic-labeled subset of FLORES-200 sentences, also improves (-8.5%), while MGSM [Shi et al., 2023] is effectively unchanged (+0.1%), as expected for its small set of languages, all of which appear in the 101-language mC4 corpus. Appendix Table D.12 reports the full 16K–256K sweep for all four corpora, including the 32K row summarized here. A scan of all mC4 training shards (held-out evaluation shards excluded) finds no document identical to a FLORES-200 sentence among 89.47M documents in 101 languages; a 50-character prefix-substring scan finds 53 FLORES-200 overlaps among 201,686 eligible sentences (Appendix C.3).

Alphabet separation trade-off. UBE's dual alphabet separates the byte identities of the UTF-8 and UTF-16 paths, so it can lose some shared-byte compression relative to BBPE. Its 512-symbol initial alphabet also consumes 256 merge slots that standard BBPE uses for subword merges. The `En-only` regime tests the worst-case cost: with only English training data, UBE wins just 37 of 204 languages

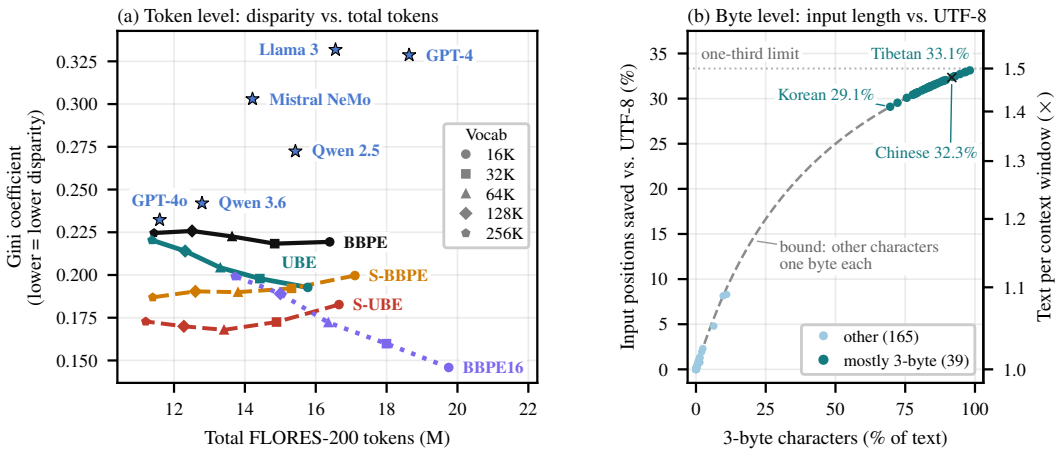

Figure 2: (a) Token-premium Gini vs. total FLORES-200 token count (`101-lang`; 16K–256K vocabularies). S-UBE and S-BBPE apply SuperBPE-style boundary relaxation to UBE and BBPE, respectively. Public LLM tokenizers (stars) are reference points (Appendix Table H.2); lower-left is better. (b) Byte-level input positions saved by UBE relative to UTF-8, measured before BPE merging, for each of the 204 languages in FLORES-200 (Appendix Table I.7; "mostly 3-byte": at least half of the characters take three bytes in UTF-8; "Chinese": mean of Chinese (Simplified), Chinese (Traditional), and Cantonese); the dashed upper bound assumes all remaining characters use one byte.

at 16K, because most languages cannot recoup the lost merge slots. Two observations mitigate this. First, typical regressions are small: at `En-only/16K`, the median increase among non-English regressions is just 0.090 tokens per sentence (mean 0.692), while UBE's token-count reductions on scripts that remain near the encoding floor reach up to 164.936 tokens per sentence. Second, the Gini reduction at `En-only/16K` (-13.9%) confirms that UBE concentrates its benefits precisely where disparity is largest. The coverage threshold for the PUA alphabet is low: moving from `En-only` to just 6 languages spanning Latin text and several 3-byte BMP script families jumps UBE's per-language wins from 37/204 to 202/204 at 16K, so even modest script coverage lets the PUA alphabet learn productive merges that offset the alphabet overhead. UBE wins 191/204 languages at `6-lang/32K` and 189/204 at `101-lang/32K`. When retokenizing the training word table at `101-lang/32K`, UBE uses nearly as much of its vocabulary as BBPE (active utilization 99.8% vs. 99.9%), never-emitted merge tokens fall from 13 to 9, and the share of merges that join two base symbols rises from 7.9% to 17.3% (Appendix D.5.3). Word-level token composition and decomposition examples are collected in Appendix D.5.1.

Gini vs. absolute token counts. Gini alone does not capture absolute token cost: BBPE16, designed for CJK-heavy ASR, achieves the lowest Gini (0.1598) because its uniform 2-byte floor narrows relative premiums, but at the cost of higher absolute token counts for English (31.160→41.317) in mixed-script LM settings. At the 32K main-text point (Figure 2a; full numerical breakdown in Appendix Table D.3), UBE reduces Gini while also lowering the overall average token count by 2.8% (71.889→69.857). SuperBPE-style boundary relaxation [Liu et al., 2025] is complementary to

Table 2: Token-premium Gini on four corpora (`101-lang/32K`); UDHR is a public-text stress test with prefix overlaps.

Dataset	Units	UBE	BBPE	ΔGini
FLORES-200	204	**0.1979**	0.2183	-9.4%
UDHR	387	**0.2221**	0.2499	-11.1%
SIB-200	205	**0.2102**	0.2297	-8.5%
MGSM	11	0.1843	**0.1841**	+0.1%

Table 3: Representative per-language token counts on FLORES-200 (`101-lang/32K`; mean tokens per sentence); other vocabulary sizes in Appendix Table C.3.

Language	Script	UBE	BBPE	Δ%
Santali	Ol Chiki	**222.944**	341.287	-34.7%
Central Atlas Tamazight	Tifinagh	**183.305**	269.166	-31.9%
Tibetan	Tibetan	**286.737**	336.804	-14.9%
Arabic	Arabic	**60.407**	60.600	-0.3%
English	Latin	**31.068**	31.160	-0.3%

UBE but shifts the trade-off: S-BBPE lowers Gini further at 32K (UBE 0.1979 vs. S-BBPE 0.1922) while inflating English tokens (31.160→33.143), and S-UBE lowers Gini further still while plain UBE keeps fewer total FLORES-200 tokens at 16K–64K. Unlike these alternatives, UBE reduces English token counts at every tested vocabulary size on FLORES-200 in the main multilingual settings (`6-lang` and `101-lang`; `En-only` excluded); it also lowers LC-BMP counts at each size while keeping CJK counts within 1% of BBPE's, and it has the lowest total FLORES-200 count among the standalone methods at every tested vocabulary size. UBE keeps ASCII bytes but writes each 3-byte BMP character as two base symbols, so BPE spends fewer merges on such characters and more on ASCII subwords, lowering English counts (Appendix D.5.1). Appendix Tables D.3 and D.4 report the full 16K–256K standalone and composition sweeps.

Mixed-script text and representative languages. On 400 naturally occurring mixed-script mC4 documents (Appendix C.2), UBE uses fewer tokens than BBPE in 7 of 8 tested languages, with Bengali showing a small regression (+0.4%); BBPE16 applies the same uniform 2-byte floor to both CJK-heavy and already-efficient spans in this slice, so its aggregate token count is 6.6% above BBPE. The same pattern holds in the primary `101-lang/32K` setting: UBE cuts token counts sharply for low-coverage 3-byte scripts such as Santali, Central Atlas Tamazight, and Tibetan, while token counts also decrease for English and Arabic (Table 3). On 100 SWE-bench tasks and their ten released MAPS translations [Jimenez et al., 2024, Hofman et al., 2026], UBE uses fewer tokens than BBPE in all 11 languages, with or without the edited code files (which make translated inputs 99.24% ASCII bytes; Appendix E.6). A complementary 6-language stress-set slice appears in Appendix C.1; full per-language values, including the small regression cases, appear in Appendix I.2.

3.2 Language model evaluation

Language model protocol. To connect intrinsic disparity to downstream behavior, we train matched LM comparisons over {UBE, BBPE} × {`En-only`, `6-lang`, `101-lang`}; within each setting, the same 101-language mC4 corpus, architecture, optimizer, and token-step budget are held fixed while only the tokenizer changes. The primary setting is `101-lang/32K`, where UBE and BBPE are exposure-matched to within $< 0.3\%$ raw-text coverage; lower-coverage rows are fixed-compute stress tests where tokenizer quality and raw-text exposure co-vary. Among our LM vocabularies, the intrinsic UBE–BBPE gap is largest at 32K; larger-vocabulary variants, a 1.3B scale check, and SuperBPE composition runs appear in Appendix F.1, with training details in Appendix B.2.

We evaluate held-out per-language bits per byte (BPB) on deterministic mC4 terminal shards and zero-shot accuracy on multilingual and English benchmarks. BPB always uses the raw UTF-8 byte count of the underlying text as the denominator for both tokenizer conditions, so UBE and BBPE likelihoods are normalized on the same content scale. Exact model sizes, token budgets, optimizer settings, sequence length, BPB sampling, and evaluation splits are reported in Appendix B.2. Appendix Table H.1 situates the reported suite against recent tokenizer-comparison papers, and Appendix H.2 surveys compact public LM tokenizer scales.

LM results (182M scale). Table 4 summarizes the matched 182M/32K LM comparison across the three tokenizer-coverage regimes.

Table 4: LM evaluation (182M, 32K vocabulary, 200K mC4 steps; primary seed; multi-seed in Appendix F.2) across three *tokenizer-training* configurations. Every LM is trained on the same 101-language mC4 corpus; En-only/6-lang/101-lang identify only the corpus used to learn the tokenizer vocabulary. Bits per byte (BPB) and BPB Gini use 101 held-out mC4 language splits; benchmark columns report zero-shot accuracy (Bele = Belebele, HSwag = HellaSwag, LAMB = LAMBADA).

Config	Tok	BPB↓	BPB Gini↓	XNLI	Bele	XCOPA	ARC-E	ARC-C	HSwag	PIQA	LAMB
En-only	UBE	**1.4698**	**0.1989**	**34.4**	23.0	**51.5**	**38.5**	17.0	**27.8**	**62.7**	22.5
	BBPE	1.4909	0.1993	34.3	**23.6**	51.4	38.1	**18.3**	27.5	61.9	**23.6**
6-lang	UBE	1.3760	**0.1880**	**34.9**	23.2	**52.2**	**36.7**	**18.4**	**28.0**	**62.0**	**23.1**
	BBPE	**1.3680**	0.1930	34.6	**23.5**	52.0	35.6	17.1	27.4	61.3	**23.1**
101-lang	UBE	1.5237	**0.1739**	**34.4**	23.7	51.8	**35.9**	**17.8**	**27.9**	**60.1**	**23.1**
	BBPE	**1.5174**	0.1762	34.3	**24.8**	**52.0**	35.5	17.0	27.7	59.3	22.8

In the primary seed, UBE has lower BPB Gini than BBPE in all three coverage regimes. Across 3 seeds at `101-lang/32K`, however, the mean BPB Gini is effectively tied (UBE 0.1822±0.0074 vs. BBPE 0.1819±0.0052; Appendix Table F.5), with cross-seed variability comparable to the gap. We therefore do not claim a systematic LM-level BPB-dispersion advantage, and we treat the matched end-of-training BPB dispersion as neutral. Across macro BPB, multilingual benchmarks, and English benchmarks, deltas are small and mixed.

BPB breakdown and scale checks. Disaggregated by encoding-floor group, held-out BPB differences are small in the median: median ΔBPB is +0.0064 for 1-byte, +0.0070 for 2-byte, and +0.0029 for 3-byte scripts. Larger language-specific shifts occur in both directions (Lao improves; Burmese and Sinhala regress), but only Lao's shift remains large in all 3 seeds (Appendix F.2). We report BPB Gini as the primary cross-lingual dispersion measure; absolute per-language BPB values (from which any non-ratio dispersion statistic can be recomputed) are in Appendix Table I.6. At 64K and 128K, intrinsic gaps narrow as merge coverage improves. Held-out BPB and downstream benchmark differences stay small and mixed across the 64K, 128K, and 1.3B/32K scale checks (Appendix F.1), consistent with prior evidence that tokenizer compression or token-count changes need not directly predict downstream LM quality [Schmidt et al., 2024]. Full per-language XNLI, XCOPA, and BPB tables (`101-lang/32K`) appear in Appendix I.3.

Source-byte-matched LM comparison. With the primary `101-lang/32K` UBE/BBPE tokenizers, we compare UBE, BBPE, and BBPE16 at matched supervised UTF-8 byte exposure and on identical held-out text. UBE has the lowest common-text BPB and BPB Gini in this three-way comparison; task accuracy is mixed (Appendix Table E.8).

Context-window and prompt-processing effects. Token-count reductions translate into fixed-budget capacity and prefill effects: with usable-context gain under a fixed token budget defined as $m_{\mathrm{BBPE}}(\ell)/m_{\mathrm{UBE}}(\ell) - 1$, Santali (Ol Chiki) gains 53% more usable context at `101-lang/32K`, with English also gaining 0.3%; in a separate `6-lang/32K` same-content prompt-processing benchmark, full-model prefill rate improves by +45.0% for Amharic and +3.9% for English (Appendix Table F.6). We time the full model rather than isolating self-attention because tokenizer-induced length changes propagate through multilayer perceptron (MLP) and memory-bandwidth costs as well.

3.3 Composition and tokenizer comparisons

UBE acts as a reusable representation layer: Table 5 brings together matched substitutions under standard, SuperBPE-style, and SCRIPT-style boundaries and MYTE's morphology maps. In each comparison, UBE lowers the English, LC-BMP, and overall mean sequence lengths while preserving the complementary component.

Table 5: Composition with UBE on FLORES-200 for 101-language/32K BPE vocabularies trained on the full primary corpus or on the 50 MiB/language prefix. Values are mean BPE tokens per sentence; in the MYTE row, the UBE and BBPE columns report MYTE-UBE and MYTE base symbols. The CJK and LC-BMP groups are defined in Appendix D.2. Gini uses 203 non-English premiums.

	English		CJK		LC-BMP		Overall (204)		Gini	
Fixed component	UBE	BBPE	UBE	BBPE	UBE	BBPE	UBE	BBPE	UBE	BBPE
Primary corpus										
Standard	**31.068**	31.160	**42.418**	42.444	**219.709**	275.681	**69.857**	71.889	**0.1979**	0.2183
SuperBPE-style	**32.973**	33.143	**43.496**	43.595	**192.162**	245.653	**72.137**	74.187	**0.1725**	0.1922
SCRIPT-style	**31.652**	31.698	42.104	**42.054**	**228.269**	286.663	**68.001**	69.957	**0.1824**	0.2036
50 MiB/language prefix										
Standard	**36.938**	36.959	**57.385**	58.956	**222.054**	272.617	**66.079**	67.875	**0.2086**	0.2281
SuperBPE-style	**39.524**	39.586	**58.576**	60.871	**189.674**	236.155	**67.208**	68.858	**0.1736**	0.1916
SCRIPT-style	**37.539**	37.569	**56.741**	58.133	**223.497**	281.154	**63.918**	65.677	**0.1832**	0.2048
MYTE maps	**109.094**	109.140	**94.610**	124.612	**247.622**	349.398	**141.354**	148.142	**0.0925**	0.1177

UBE's effect on CJK token counts varies with how fully BBPE's learned merges cover CJK characters. At a vocabulary size of 32K and across three boundary policies (Appendix E.7), BBPE splits 0.2–0.6%

of Hangul, Han, and kana occurrences into three or more pieces under the full primary corpus, and UBE's CJK mean token count remains within 0.3% of BBPE's. Under the 50 MiB/language cap, this fragmentation rate rises to 3.5–5.5%, and UBE reduces the CJK mean token count by 2.4–3.8%. These results are consistent with a coverage-based explanation, which the fixed-volume mixtures below test directly.

Composition with boundary policies. SuperBPE-style boundary relaxation [Liu et al., 2025] and SCRIPT-style pre-tokenization [Land and Arnett, 2025] act on a different component from UBE's byte representation. The SCRIPT-style comparison shares the official script/category split but retains UBE/BBPE symbols and standard, unconstrained BPE merges. Compared with standard pre-tokenization, SCRIPT-style achieves a lower overall mean primarily through reductions on South Asian scripts (Appendix Table E.2). Replacing BBPE with UBE also lowers Gini under all three boundary policies (Appendix E.2). The same pattern holds with a 50 MiB/language training cap (Appendix Tables E.3 and E.5).

Composition with morphology maps and byte-level inputs. Keeping MYTE's released decomposition and morphology-merge maps fixed [Limisiewicz et al., 2024], we route only residual characters through UBE. MYTE-UBE preserves MYTE's decoded output while reducing total base-symbol positions on FLORES-200 by 4.58%, CJK means by 24.08%, and LC-BMP means by 29.13% (201/204 languages lower, three tied, none higher; Appendix Table E.6). These are pre-BPE base symbols, not learned BPE tokens. Byte-level models such as ByT5 [Xue et al., 2022] use one input position per base symbol. Against UTF-8 bytes, UBE alone shortens mean CJK and LC-BMP inputs by 31.73% and 31.87% and reduces total positions by 10.21%; inputs are shorter in 202/204 languages and longer in none (the English mean is 0.04% lower because 3-byte characters such as em dashes and curly quotes take two symbols instead of three), and byte-level Gini falls from 0.2077 to 0.1500 (28% lower, against 6–14% in the core BPE comparisons). Mean inputs in each of the 39 languages written mostly in 3-byte characters (at least half of their characters) are 29.10–33.13% shorter, near the one-third maximum, so the same context window holds 1.41–1.50 times as much of this text (Figure 2b; Appendix E.2).

Comparison with SCRIPT-BPE. In the separate five-tokenizer comparison in Appendix Table E.3 (50 MiB/language), official SCRIPT-BPE attains the lowest overall mean and token-premium Gini. UBE uses fewer English tokens and fewer tokens in 169/204 languages (Appendix Table E.4). SCRIPT-BPE's gains concentrate in specific script cohorts; this pattern persists across vocabulary sizes, larger training prefixes, and XNLI/XCOPA/MGSM inputs (Appendix E.2).

Sensitivity to the tokenizer-training corpus. The two corpus blocks of Table 5 compare tokenizers trained on the full primary corpus and on the 50 MiB/language prefix. Across the six composition variants, the capped tokenizers use 5.4–7.2% fewer tokens overall, but 18.5–19.9% more for English and 34.7–39.6% more for CJK, while the LC-BMP mean changes by at most 3.9%. The cap lowers the training share of English from 29.2% to 1.0%, that of Korean, Chinese, and Japanese from 26.5% to 3.0%, and that of Hindi and Thai from 17.4% to 2.0%, so their text receives fewer of the frequency-ordered BPE merges [Conneau and Lample, 2019, NLLB Team, 2024, Hayase et al., 2024]. The share of the remaining 95 training languages rises from 26.8% to 94.1%, which shortens token sequences for the many non-English Latin-, Arabic-, and Cyrillic-script languages that dominate the equally weighted FLORES-200 average, as flatter vocabulary-sampling mixes do for low-resource languages in multilingual translation [Arivazhagan et al., 2019]; most scripts in the LC-BMP group are not used by any training language. Within the capped condition, replacing BBPE with UBE keeps English counts lower under every boundary policy. UBE also lowers Gini in both corpus regimes. Appendix E.7 reports all script groups and paired intervals; the five-tokenizer LM comparison appears in Appendix Table E.9.

Emulated commercial tokenizer-training mixtures. To vary language composition at a fixed volume, we train UBE and BBPE at 32K–200K on 30 GiB tokenizer-training corpora that emulate the mixtures inferred for GPT-4o and Mistral NeMo [Hayase et al., 2024], plus a Chinese-centric scenario. In all ten pairs, including a second corpus sample, UBE lowers total and LC-BMP tokens and token-premium Gini while also lowering English tokens (Appendix E.8).

4 Related work

Tokenized length varies sharply across languages on parallel text [Petrov et al., 2023], and concurrent and later work links this disparity to API pricing, utility, downstream performance, latency, and access costs [Ahia et al., 2023, Lundin et al., 2026, Dixit and Dixit, 2026, Teklehaymanot and Nejdl, 2025]; Appendix H.3 reviews disparity metrics and analyses of its causes.

Mitigations fall into three groups. Within-substrate methods change merge or segmentation objectives, pre-tokenization, vocabulary allocation, or post-hoc vocabulary additions while keeping the byte encoding fixed [Liu et al., 2025, Schmidt et al., 2025, Abagyan et al., 2026, Churchill and Skiena, 2026]; these can compose with UBE but leave the UTF-8 encoding floor unchanged. Substrate-changing methods alter the input symbols: BBPE16 applies UTF-16 uniformly [Kim et al., 2026], ERNIE 5.0 encodes text in UTF-16BE for byte-level fallback [ERNIE Team, 2026], SCRIPT-BPE combines a script-aware representation with script/category boundaries and constrained merges [Land and Arnett, 2025], MYTE learns morphology-driven byte codes [Limisiewicz et al., 2024], and Bit-level BPE re-splits the byte stream at the bit level [Moon et al., 2025]. UBE instead routes each character between UTF-8 and UTF-16 inside one reversible tokenizer; Section 3.3 and Appendix E compare these representation, boundary, and morphology-based alternatives at the tokenizer and model levels. Tokenization-free and byte-level models operate directly on characters or bytes: CANINE [Clark et al., 2022], ByT5 [Xue et al., 2022], BLT [Pagnoni et al., 2025], MAGNET [Ahia et al., 2024], and H-Net [Hwang et al., 2026]; they reduce reliance on fixed tokenizers but change the model's architecture or input granularity, whereas UBE preserves standard BPE training and exact decoding. In follow-up work [Kim, 2026], small causal predictors with one-symbol lookahead predict grapheme boundaries in Brahmic scripts substantially more accurately on UBE input than on UTF-8 input; Appendix H.3 reports these results and surveys further byte-level architectures.

5 Conclusion

UBE replaces BBPE's single UTF-8 substrate with exactly reversible per-character UTF-8/UTF-16 routing while keeping standard BPE intact; the routing itself requires neither language identification nor a Unicode script table. It reduces token-budget disparity across the main intrinsic evaluations, especially at 16K–64K (FLORES-200 Gini 6–14% lower), also lowers English token counts in the main multilingual settings, and composes with SuperBPE-style and SCRIPT-style boundary policies and MYTE's morphology maps while preserving each complementary mechanism. For byte-level models, UBE shortens inputs by 29–33% in every FLORES-200 language written mostly in 3-byte characters, lengthens no input, and lowers byte-level Gini by 28%. Downstream differences remain small and mixed; in the source-byte-matched BBPE/BBPE16 comparison, UBE has the lowest common-text BPB and BPB Gini, while same-content prompt processing improves by up to +45.0%. Under emulated commercial tokenizer-training mixtures at 32K–200K, UBE still lowers Gini and the total, LC-BMP, and English token counts. UBE is most useful for byte-level models, compact vocabularies, and LC-BMP languages near the encoding floor.

Limitations. UBE addresses only the substrate-level source of token-budget disparity; per-language vocabulary sizing, data balance, pre-tokenization, and merge-rule interventions remain orthogonal. Its 256 extra PUA symbols reduce wins at very narrow coverage (`En-only/16K`: 37/204) and forgo BBPE's coincidental byte sharing across scripts, causing small regressions in a few high-coverage languages. LM evaluation covers 182M/32K, 230M/64K, 329M/128K, and 1.3B/32K settings, with 3 seeds only at `101-lang/32K`. Because UBE changes the tokenizer vocabulary, it is not a drop-in replacement for existing checkpoints. A model-level comparison includes SCRIPT-BPE and derived 32K MYTE+BPE (a 32K BPE over the MYTE representation; Appendix E.5); the released 256-symbol MYTE itself and Bit-level BPE remain future work. Appendix H.4 covers downstream coverage for the most-affected scripts, the scope of the commercial-mixture analysis, constrained decoding, and broader impacts.

Acknowledgments and Disclosure of Funding

Hyunsik Kim thanks his wife, Jiyoung Kim, and his daughter, Nahyun Kim, for all their support and understanding throughout this work. He also appreciates Youngmoon Jung's care and thoroughness in this project, and thanks Haeri Kim and Sichen Jin at Carnegie Mellon University for inspiring the idea behind UBE. Youngmoon Jung is grateful to Hyunsik Kim for the great research opportunity and dedicated collaboration.

This research received no funding. The authors declare no competing interests beyond the affiliation shown above.

References

Diana Abagyan, Alejandro R. Salamanca, Andres Felipe Cruz-Salinas, Kris Cao, Hangyu Lin, Acyr Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. One tokenizer to rule them all: Emergent language plasticity via multilingual tokenizers. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3118–3136, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.141.

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, Dongdong Chen, Junkun Chen, Weizhu Chen, Yen-Chun Chen, Yi-ling Chen, Qi Dai, Xiyang Dai, Ruchao Fan, Mei Gao, et al. Phi-4-Mini technical report: Compact yet powerful multimodal language models via mixture-of-LoRAs. *arXiv preprint arXiv:2503.01743*, 2025. doi: 10.48550/arXiv.2503.01743.

David Ifeoluwa Adelani, Hannah Liu, Xiaoyu Shen, Nikita Vassilyev, Jesujoba O. Alabi, Yanke Mao, Haonan Gao, and En-Shiun Annie Lee. SIB-200: A simple, inclusive, and big evaluation dataset for topic classification in 200+ languages and dialects. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 226–245, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-long.14.

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, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.614.

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah Smith. MAGNET: Improving the multilingual fairness of language models with adaptive gradient-based tokenization. In *Advances in Neural Information Processing Systems*, volume 37, pages 47790–47814. Curran Associates, Inc., 2024. doi: 10.52202/079017-1514.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martin Blazquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Agustín Piqueres Lajarín, Hynek Kydlíček, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan Son Nguyen, Ben Burtenshaw, Clémentine Fourrier, Haojun Zhao, Hugo Larcher, Mathieu Morlon, Cyril Zakka, Colin Raffel, Leandro Von Werra, and Thomas Wolf. SmolLM2: When smol goes big — data-centric training of a fully open small language model. In *Second Conference on Language Modeling*, 2025.

Sawsan Alqahtani, Mir Tafseer Nayeem, Md Tahmid Rahman Laskar, Tasnim Mohiuddin, and M Saiful Bari. Stop taking tokenizers for granted: They are core design decisions in large language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8432, Rabat, Morocco, March 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.394.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*, 2019. doi: 10.48550/arXiv.1907.05019.

Catherine Arnett, Tyler A. Chang, and Benjamin Bergen. A bit of a problem: Measurement disparities in dataset sizes across languages. In *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages @ LREC-COLING 2024*, pages 1–9, Torino, Italia, May 2024. ELRA and ICCL.

Catherine Arnett, Tyler Chang, Stella Biderman, and Benjamin Bergen. Explaining and mitigating crosslingual tokenizer inequities. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 63277–63313. Curran Associates, Inc., 2025. doi: 10.52202/085713-2126.

Ander Artola Velasco, Stratis Tsirtsis, Nastaran Okati, and Manuel Gomez Rodriguez. Is your LLM overcharging you? Tokenization, transparency, and incentives. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 123628–123652. PMLR, 06–11 Jul 2026.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. The Belebele benchmark: a parallel reading comprehension dataset in 122 language variants. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 749–775, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.44.

Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, Meng Lee, Emad Mostaque, Michael Pieler, Nikhil Pinnaparju, Paulo Rocha, Harry Saini, Hannah Teufel, Niccolo Zanichelli, and Carlos Riquelme. Stable LM 2 1.6B technical report. *arXiv preprint arXiv:2402.17834*, 2024. doi: 10.48550/arXiv.2402.17834.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR, 23–29 Jul 2023.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, et al. BLOOM: A 176B-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2023. doi: 10.48550/arXiv.2211.05100.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

Maharaj Brahma, N J Karthika, Atul Kumar Singh, Devaraja Adiga, Smruti Bhate, Ganesh Ramakrishnan, Rohit Saluja, and Maunendra Sankar Desarkar. MorphTok: Morphologically grounded tokenization for Indic languages. In *Tokenization Workshop*, 2025.

Geoffrey Churchill and Steven Skiena. Reducing tokenization premiums for low-resource languages. *arXiv preprint arXiv:2601.13328*, 2026. doi: 10.48550/arXiv.2601.13328.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. CANINE: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022. doi: 10.1162/tacl_a_00448.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. doi: 10.48550/arXiv.1803.05457.

Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32, pages 7059–7069. Curran Associates, Inc., 2019.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359. Curran Associates, Inc., 2022.

Kusal Darshana. Separate before you compress: The WWHO tokenization architecture. *arXiv preprint arXiv:2603.25309*, 2026. doi: 10.48550/arXiv.2603.25309.

Mark Davis and Ned Holbrook. Unicode technical standard #51: Unicode emoji, version 17.0. https://www.unicode.org/reports/tr51/tr51-29.html, 2025.

Aradhya Dixit and Shreem Dixit. The script tax: Measuring tokenization-driven efficiency and latency disparities in multilingual language models. *arXiv preprint arXiv:2602.11174*, 2026. doi: 10.48550/arXiv.2602.11174.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig, editors. *Ethnologue: Languages of the World*. SIL International, Dallas, Texas, 27th edition, 2024.

ERNIE Team. ERNIE 5.0 technical report. *arXiv preprint arXiv:2602.04705*, 2026. doi: 10.48550/arXiv.2602.04705.

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 *Second Conference on Language Modeling*, 2025.

Andy Georges, Dries Buytaert, and Lieven Eeckhout. Statistically rigorous Java performance evaluation. In *Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 57–76. ACM, October 2007. doi: 10.1145/1297027.1297033.

Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. Unpacking tokenization: Evaluating text compression and its correlation with model performance. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2274–2286, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.134.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783.

Jonathan Hayase, Alisa Liu, Yejin Choi, Sewoong Oh, and Noah Smith. Data mixture inference attack: BPE tokenizers reveal training data compositions. In *Advances in Neural Information Processing Systems*, volume 37, pages 8956–8983. Curran Associates, Inc., 2024. doi: 10.52202/079017-0285.

Torsten Hoefler and Roberto Belli. Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 73:1–73:12, Austin, TX, USA, November 2015. ACM. ISBN 978-1-4503-3723-6. doi: 10.1145/2807591.2807644.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pages 30016–30030. Curran Associates, Inc., 2022.

Omer Hofman, Jonathan Brokman, Oren Rachmil, Shamik Bose, Vikas Pahuja, Toshiya Shimizu, Trisha Starostina, Kelly Marchisio, Seraphina Goldfarb-Tarrant, and Roman Vainshtein. MAPS: A multilingual benchmark for agent performance and security. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 821–845. Association for Computational Linguistics, 2026.

Songbo Hu, Ivan Vulić, and Anna Korhonen. Quantifying language disparities in multilingual large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4003–4018, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.199.

HuggingFaceTB. SmolLM3-3B model card and configuration. https://huggingface.co/HuggingFaceTB/SmolLM3-3B, 2025.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling. In *The Fourteenth International Conference on Learning Representations*, 2026.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Julie Kallini, Shikhar Murty, Christopher D. Manning, Christopher Potts, and Róbert Csordás. MrT5: Dynamic token merging for efficient byte-level language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Vani Kanjirangat, Tanja Samardzic, Ljiljana Dolamic, and Fabio Rinaldi. Tokenization and representation biases in multilingual models on dialectal NLP tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 23992–24010, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1224.

Hyunsik Kim. The right byte first: How UTF-16 helps multilingual boundary prediction. *arXiv preprint arXiv:2610.22071*, 2026. doi: 10.48550/arXiv.2610.22071.

Hyunsik Kim, Haeri Kim, Munhak Lee, and Kyungmin Lee. BBPE16: UTF-16-based byte-level byte-pair encoding for improved multilingual speech recognition. In *ICASSP 2026 – 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 17067–17071. IEEE, 2026. doi: 10.1109/ICASSP55912.2026.11460641.

Zhenglun Kong, Yize Li, Fanhu Zeng, Lei Xin, Shvat Messica, Xue Lin, Pu Zhao, Manolis Kellis, Hao Tang, and Marinka Zitnik. Token reduction should go beyond efficiency in generative models – from vision, language to multimodality. *arXiv preprint arXiv:2505.18227*, 2026. doi: 10.48550/arXiv.2505.18227.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012.

Sander Land and Catherine Arnett. BPE stays on SCRIPT: Structured encoding for robust multilingual pretokenization. In *Tokenization Workshop*, 2025.

Tomasz Limisiewicz, Terra Blevins, Hila Gonen, Orevaoghene Ahia, and Luke Zettlemoyer. MYTE: Morphology-driven byte encoding for better and fairer multilingual language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers), pages 15059–15076, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.804.

Alisa Liu, Jonathan Hayase, Valentin Hofmann, Sewoong Oh, Noah A. Smith, and Yejin Choi. SuperBPE: Space travel for language models. In *Second Conference on Language Modeling*, 2025.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32431–32454. PMLR, 21–27 Jul 2024.

Jessica M. Lundin, Ada Zhang, Nihal Karim, Hamza Louzan, Guohao Wei, David Ifeoluwa Adelani, and Cody Carroll. The token tax: Systematic bias in multilingual tokenization. In *Proceedings of the 7th Workshop on African Natural Language Processing (AfricaNLP 2026)*, pages 103–112, Rabat, Morocco, March 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.africanlp-main.10.

Meta. Llama 3.2 model card. https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md, 2024.

Microsoft. Phi-3.5-mini-instruct model card. https://huggingface.co/microsoft/Phi-3.5-mini-instruct, 2024.

Microsoft. Phi-4-mini-instruct model card. https://huggingface.co/microsoft/Phi-4-mini-instruct, 2025.

Benjamin Minixhofer, Tyler Murray, Tomasz Limisiewicz, Anna Korhonen, Luke Zettlemoyer, Noah A. Smith, Edoardo M. Ponti, Luca Soldaini, and Valentin Hofmann. Retrofitting language models to operate over bytes. *Nature*, 2026. doi: 10.1038/s41586-026-11111-4.

Mistral AI. Mistral NeMo. https://mistral.ai/news/mistral-nemo/, 2024a.

Mistral AI. Mistral-Nemo-Base-2407 model card and configuration. https://huggingface.co/mistralai/Mistral-Nemo-Base-2407, 2024b.

Sangwhan Moon, Tatsuya Hiraoka, and Naoaki Okazaki. Bit-level BPE: Below the byte boundary. *arXiv preprint arXiv:2506.07541*, 2025. doi: 10.48550/arXiv.2506.07541.

Mir Tafseer Nayeem, Sawsan Alqahtani, Md Tahmid Rahman Laskar, Tasnim Mohiuddin, and M Saiful Bari. Beyond fertility: Analyzing STRR as a metric for multilingual tokenization evaluation. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025.

NLLB Team. Scaling neural machine translation to 200 languages. *Nature*, 630:841–846, 2024. doi: 10.1038/s41586-024-07335-x.

OpenAI. tiktoken. https://github.com/openai/tiktoken, 2023.

Abraham Toluwase Owodunni, Orevaoghene Ahia, and Sachin Kumar. FLEXITOKENS: Flexible tokenization for evolving language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 17170–17190, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.848.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E. Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.453.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems*, volume 36, pages 36963–36990. Curran Associates, Inc., 2023.

Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. XCOPA: A multilingual dataset for causal commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2362–2376, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.185.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems*, volume 5, pages 606–624, 2023.

Qwen Team. Qwen2.5: A party of foundation models. https://qwenlm.github.io/blog/qwen2.5/, September 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. doi: 10.48550/arXiv.2412.15115.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model. https://qwen.ai/blog?id=qwen3.6-27b, April 2026.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Souvik Rana, Arul Menezes, Ashish Kulkarni, Chandra Khatri, and Shubham Agarwal. MUTANT: A recipe for multilingual tokenizer design. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 46261–46277, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.2146.

Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, Ramesh Chukka, Cody Coleman, Sam Davis, Pan Deng, Greg Diamos, Jared Duke, Dave Fick, J. Scott Gardner, Itay Hubara, Sachin Idgunji, Thomas B. Jablin, Jeff Jiao, Tom St. John, Pankaj Kanwar, David Lee, Jeffery Liao, Anton Lokhmotov, Francisco Massa, Peng Meng, Paulius Micikevicius, Colin Osborne, Gennady Pekhimenko, Arun Tejusve Raghunath Rajan, Dilip Sequeira, Ashish Sirasao, Fei Sun, Hanlin Tang, Michael Thomson, Frank Wei, Ephrem Wu, Lingjie Xu, Koichi Yamada, Bing Yu, George Yuan, Aaron Zhong, Peizhao Zhang, and Yuchen Zhou. MLPerf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 446–459, May 2020. doi: 10.1109/ISCA45697.2020.00045.

Varshini Reddy, Craig W. Schmidt, Yuval Pinter, and Chris Tanner. How much is enough? The diminishing returns of tokenization training data. In *Tokenization Workshop*, 2025.

Craig W. Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. Tokenization is more than compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 678–702, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.40.

Craig W. Schmidt, Varshini Reddy, Chris Tanner, and Yuval Pinter. Boundless byte pair encoding: Breaking the pre-tokenization barrier. In *Second Conference on Language Modeling*, 2025.

Chen Shani, Yuval Reif, Nathan Roll, Dan Jurafsky, and Ekaterina Shutova. The roots of performance disparity in multilingual language models: Intrinsic modeling difficulty or design choices? *arXiv preprint arXiv:2601.07220*, 2026. doi: 10.48550/arXiv.2601.07220.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019. doi: 10.48550/arXiv.1911.02150.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual Chain-of-Thought reasoners. In *The Eleventh International Conference on Learning Representations*, 2023.

Kevin Slagle. SpaceByte: Towards deleting tokenization from large language modeling. In *Advances in Neural Information Processing Systems*, volume 37, pages 124925–124950. Curran Associates, Inc., 2024. doi: 10.52202/079017-3967.

Hailay Kidu Teklehaymanot and Wolfgang Nejdl. Tokenization disparities as infrastructure bias: How subword systems create inequities in LLM access and efficiency. In *2025 3rd International Conference on Foundation and Large Language Models (FLLM)*, pages 822–828. IEEE, November 2025. doi: 10.1109/FLLM67465.2025.11390956.

Aamod Thakur, Ajay Nagpal, Atharva Savarkar, Kundeshwar Pundalik, Siddhesh Dosi, Piyush Sawarkar, Viraj Thakur, Rohit Saluja, Maunendra Sankar Desarkar, and Ganesh Ramakrishnan. The art of breaking words: Rethinking multilingual tokenizer design. *arXiv preprint arXiv:2508.06533*, 2025. doi: 10.48550/arXiv.2508.06533.

The Unicode Consortium. The Unicode standard, version 17.0.0. https://www.unicode.org/versions/Unicode17.0.0/, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

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, January 2025. Association for Computational Linguistics.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. Neural machine translation with byte-level subwords. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9154–9160, 2020. doi: 10.1609/aaai.v34i05.6451.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022. doi: 10.1162/tacl_a_00461.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. GLM-130B: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations*, 2023.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. TinyLlama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024. doi: 10.48550/arXiv.2401.02385.

A Universal byte-level encoding (UBE) algorithm and correctness

A.1 Pre-tokenizer: UTF-8-byte-length routing

The pre-tokenizer applies three stages sequentially:

Stage 1: GPT-2 regex splitting. UBE uses the same GPT-2 byte-level pre-tokenizer pattern as the BBPE baseline in this paper to split input text into word-like chunks:

```
's|'t|'re|'ve|'m|'ll|'d| ?\p{L}+| ?\p{N}+
    | ?[^\s\p{L}\p{N}]+|\s+(?!\S)|\s+
```

In the core UBE/BBPE tokenizer artifacts, the Stage-1 pre-tokenizer sets `use_regex=true` and `add_prefix_space=false`; no prefix space is inserted before regex splitting during tokenization. The tokenizer-level normalizer is `null`, so inputs are used as received, with no Unicode normalization, lowercasing, accent stripping, or compatibility folding. All tokenizer families reserve five added special tokens with IDs 0–4: `<s>`, `<pad>`, `</s>`, `<unk>`, and `<mask>`; each is serialized with `special=true` and `normalized=false`. This stage is held fixed across tokenizer families, so UBE changes the byte substrate rather than the word-boundary policy or special-token convention.

Stage 2: per-character routing. Each character is routed by a deterministic rule based only on its UTF-8 byte length. Here, "character" refers to one Unicode scalar value (the Rust `char` unit used by the implementation), not a grapheme cluster; isolated surrogate code points are outside this well-formed input domain:

- **UTF-8 path**: characters whose UTF-8 encoding is 1–2 bytes (e.g., ASCII and many Latin Extended, Cyrillic, Greek, Arabic, and Hebrew code points).
- **UTF-16 path**: characters whose UTF-8 encoding is 3–4 bytes. This route includes all 3-byte BMP scalar values and 4-byte non-BMP scalar values; only the former receive a byte-floor reduction, from 3 UTF-8 bytes to 2 UTF-16 bytes. Examples of 3-byte BMP scripts and code points include CJK characters, Devanagari, Bengali, Tamil, Thai, Georgian, Tibetan, Ethiopic, Myanmar, and Khmer.

Figure A.1 compares the UTF-8 and UTF-16 bit layouts of each length class.

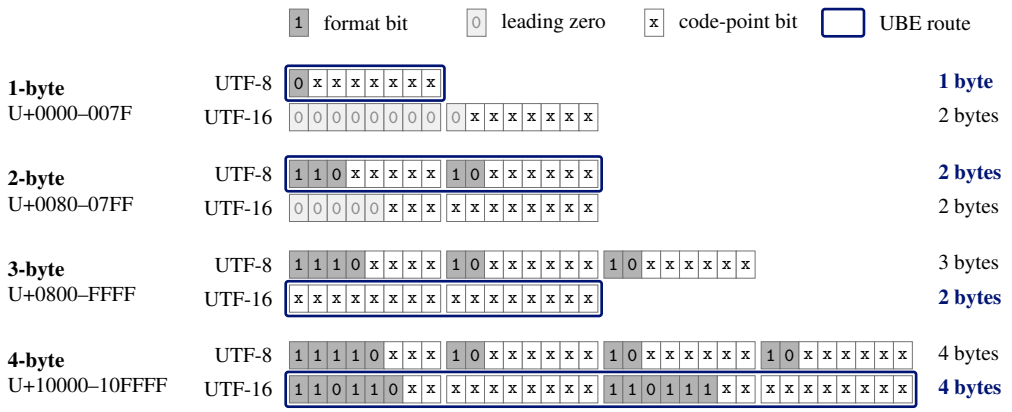

Figure A.1: Bit layouts of each UTF-8 length class under UTF-8 and UTF-16, with code units drawn most significant bit first (UBE writes them little-endian; surrogate pairs carry the code point minus `0x10000`). UTF-16 saves a byte only for 3-byte BMP code points.

Stage 3: byte serialization and alphabet mapping.

- **UTF-8 path**: character → UTF-8 bytes → GPT-2 visible-byte alphabet (256 symbols).
- **UTF-16 path**: character → UTF-16LE bytes → PUA U+E100–E1FF (256 symbols).

On the UTF-16 path, BMP code points emit 2 bytes, while non-BMP code points emit 4 bytes via surrogate pairs. The combined initial alphabet has $256 + 256 = 512$ base symbols. The two alphabets share no characters.

A.2 Reference pseudocode

The following pseudocode (Algorithms 1 and 2) summarizes the reference implementation at the level needed to reproduce the routing and exact round-trip behavior. Standard BPE merge learning, ranking, and vocabulary construction are unchanged. BPE is applied independently to each GPT-2 regex chunk, matching standard GPT-2-style BBPE and preventing learned merges from crossing pre-tokenizer chunk boundaries. Exact round-trip decoding for encoded Unicode strings is proved in Proposition 1.

Algorithm 1 UBE encode

Require: Unicode string s; trained BPE vocabulary over the 512-symbol union alphabet
Ensure: Token ID sequence

if `add_prefix_space` and s is nonempty and does not start with a space (U+0020) **then**
  $s \leftarrow$ " " $\| \, s$
end if
$chunks \leftarrow$ GPT-2-regex-split(s) ▷ Stage 1
$tokens \leftarrow [\,]$
for each chunk in $chunks$ **do**
  $chunkSymbols \leftarrow [\,]$
  for each character c in chunk **do** ▷ Stage 2: routing
    if $|\mathrm{utf8}(c)| \leq 2$ **then**
      Encode c as UTF-8 bytes; map each byte b to GPT-2$[b]$ ▷ Stage 3
    else
      Encode c as UTF-16LE bytes; map each byte b to PUA$[b]$ (U+E100+b) ▷ Stage 3
    end if
    Append mapped symbols to $chunkSymbols$
  end for
  Append BPE-merge$(chunkSymbols)$ to $tokens$ ▷ No merge crosses chunk boundaries
end for
return $tokens$

Algorithm 2 UBE decode (dual-buffer flush-on-switch)

Require: Expanded symbol sequence from BPE decoding over the mixed alphabet
Ensure: Decoded Unicode string

$B_8 \leftarrow [\,]$; $B_{16} \leftarrow [\,]$; $out \leftarrow [\,]$ ▷ buffers + output
for each symbol x in expanded BPE symbol stream **do**
  if $x \in$ GPT-2 alphabet **then**
    Flush B_{16}: decode as UTF-16LE, append to out, clear B_{16}
    Append corresponding byte to B_8
  else if $x \in$ PUA alphabet **then**
    Flush B_8: decode as UTF-8, append to out, clear B_8
    Append $(\mathrm{ord}(x) - \mathrm{0xE100})$ to B_{16}
  else ▷ special token boundary
    Flush B_8 and B_{16}; append x to out
  end if
end for
Flush remaining B_8 and B_{16}
return concat(out)

A.3 Encoding example

We complement Figure 1 (which gives the 7/9/10 base-symbol totals for "GPU性能" under UBE, BBPE, and BBPE16) with the byte-level mapping for the CJK character 性 (U+6027): BBPE encodes

it as 3 UTF-8 bytes (`0xE6, 0x80, 0xA7`), mapped to the GPT-2 alphabet, while UBE encodes it as 2 UTF-16LE bytes (`0x27, 0x60`), mapped to PUA (U+E127, U+E160).

Private-use input characters. The PUA block U+E100–E1FF is used only as an internal visible-byte alphabet for the UTF-16 path; raw input scalar values from the same Unicode range are still encoded as ordinary input characters before alphabet mapping. For example, a raw U+E100 character is a 3-byte UTF-8 scalar and is routed to the UTF-16 path. Its UTF-16LE bytes are `0x00, 0xE1`, which are represented by the two internal symbols U+E100 and U+E1E1. During decoding, these symbols are interpreted as byte values, reconstructed as the UTF-16 code unit 0xE100, and emitted as the original raw scalar value. Thus there is no separate collision case: alphabet membership identifies the byte buffer, while the reconstructed code unit determines the final Unicode scalar.

A.4 Correctness and implementation notes

Proposition 1 (Exact round-trip)**.** *Let s be a well-formed Unicode scalar-value string with no isolated surrogate code points, and let s^+ denote the string after the tokenizer's optional prefix-space preprocessing. If BPE merge tokens are expanded to base symbols before buffer routing, then $\mathit{decode}(\mathit{encode}(s)) = s^+$. In particular, $\mathit{decode}(\mathit{encode}(s)) = s$ whenever prefix-space preprocessing does not modify s.*

Proof. Prefix-space preprocessing, when enabled and applicable, deterministically replaces s with s^+ before regex splitting; the remaining argument is therefore about reconstructing s^+. BPE decoding first expands each merge token into a sequence over the two base alphabets. The encoder maps each input scalar value of s^+ into exactly one of two disjoint symbol alphabets: the visible-byte alphabet used by the UTF-8 path, and the private-use alphabet used by the UTF-16 path. Because the alphabets are disjoint, every emitted symbol uniquely identifies which byte buffer must receive it during decoding. When the symbol stream switches paths, the previously active buffer is flushed and decoded before any byte from the new path is appended, so the decoder never forms a mixed UTF-8/UTF-16 byte string. Within a same-path run, the decoder does not need to know character boundaries: the encoder wrote a concatenation of complete UTF-8 byte sequences or complete UTF-16 code-unit sequences, and the standard UTF-8 and UTF-16 decodings are bijective on well-formed byte or code-unit sequences (including surrogate-paired non-BMP scalar values), so the original scalar values are recovered. Special tokens trigger a flush of both buffers and are passed through unchanged, preserving token boundaries. Therefore every encoded span is reconstructed exactly, and the final concatenation is s^+; if prefix-space preprocessing is disabled or leaves s unchanged, this is the original string s. We additionally verified this round-trip property empirically on FLORES-200 (204 languages, 206,448 sentences, 100% pass rate). □

This proposition is a round-trip guarantee for token sequences produced by the encoder, not a claim that every arbitrary model-generated token sequence expands to a well-formed UTF-8 or UTF-16 byte string. The UBE decoder follows the usual robust byte-tokenizer convention: malformed same-path byte runs are decoded with replacement characters rather than raising an exception, so generation remains total while exactness is guaranteed only on encoder-produced streams.

The Rust core implements the routing pre-tokenizer, dual-buffer decoder, and offset post-processor, with Python wrappers exposing standard Hugging Face training APIs; the incremental training pipeline (Appendix B.1) is a separate systems layer. UBE implements the Hugging Face `PostProcessor` interface for downstream span-extraction compatibility; our configurations set `trim_offsets=false` so that leading/trailing whitespace offsets are returned as-is.

Streaming and constrained decoding audit. The decoder state needed for streaming consists only of the active alphabet tag and one pending byte buffer. The local invariant after any emitted base symbol is: all flushed output is valid Unicode, and the only unflushed bytes belong to exactly one path, either UTF-8 or UTF-16. When the next expanded symbol has the same alphabet, its byte is appended to the pending buffer; when its alphabet differs, the pending buffer is decoded and emitted before the new buffer is started. Special tokens also force both buffers to flush. Therefore cross-alphabet BPE merges do not create mixed byte strings after token expansion, and streaming implementations can expose committed output at flush points while retaining at most the current same-alphabet byte run as pending state. Guided or grammar-constrained decoders need access to the same per-symbol alphabet

tag before applying byte-level constraints; otherwise a constraint engine cannot tell whether a byte value is being interpreted as UTF-8 or UTF-16. We audit this as an implementation requirement rather than a throughput result; constrained-decoding latency is not benchmarked in this paper.

Decoding part of a token sequence. Whole-sequence decoding is exact for encoder-produced sequences (Proposition 1), and streaming with a detokenizer that retains state from the first token, such as `DecodeStream` in Hugging Face `tokenizers`, reproduced every sequence tested here. Other code paths decode only part of a sequence: incremental detokenizers that re-decode a short window of recent tokens and emit the text added by the newest ones [Firestone et al., 2025], and tools that decode slices of token ids. On the UTF-16 path, the decoder pairs bytes from the start of each same-path run. Two cases are handled by different rules. For a window or a single decode of a continuation, decoding starts at the latest token boundary that is also a character boundary and is at or before the usual start; one pass of the decoder state over the preceding tokens finds this boundary. A slice of token ids is decoded from its own start. If it ends at a character boundary, parity restores code-unit alignment: an odd number of leading UTF-16 symbols, up to the first UTF-8 symbol or the slice end, shows that the first symbol is the second byte of a code unit. That byte is replaced with U+FFFD before the remainder is decoded, and any remaining fragment of the divided character also becomes U+FFFD. On all 206,448 FLORES-200 devtest sentences, the aligned streams and continuations reproduce the expected text, and the parity rule preserves all complete characters in every tested suffix slice with the primary `101-lang/32K` UBE and BBPE tokenizers; UBE also places fewer token boundaries inside characters (7.18% vs. 9.09% of interior token positions). Both rules and per-token text are part of the released code.

B Training and evaluation pipeline

B.1 Tokenizer training pipeline

Overview. All tokenizers (UBE, BBPE, BBPE16) are trained using a two-phase distributed BPE pipeline. Phase 1 computes shard-local states once, and phase 2 merges and finalizes those states into the target vocabulary. This design separates shard-level word-count collection from merge selection, enabling reuse of shard states across vocabulary sizes for a fixed tokenizer type and fixed shard manifest.

Phase 1: shard-local state collection. Each training shard (an Arrow file containing raw text) is processed independently. For each shard, we:

1. Extract the `text` column from the Arrow file.
2. Initialize a fresh `BpeTrainer` with `vocab_size` set to the maximum target for the sweep being produced (64,000 for the core 16K/32K/64K sweep, with larger targets used for extended diagnostics) and `min_frequency=2`.
3. Run an incremental pass that accumulates pre-tokenized word counts without performing BPE merge selection. Pair statistics are derived from the retained word table during finalization.
4. Save the resulting word-count state as a JSON file.

The save-state threshold `min_frequency=2` filters per-shard singletons (word patterns appearing only once in a single shard), reducing state file size. This is a noise-reduction step, not a final vocabulary decision. Shard states are independent and can be computed in parallel across workers.

Phase 2: state merging. All shard states are merged by summing retained word counts in a single parallel reduce pass with no merge-time pruning or post-reduction filtering in the canonical path. Because addition is commutative and associative, the retained word-count totals are order-independent and exact over the retained shard states. This is not equivalent to exact whole-corpus BPE, however, because phase 1 already removes per-shard singletons: patterns appearing once in multiple shards can be lost before the reduce ever sees them. For the 101-language configuration this canonical merge retains approximately 139M unique word patterns after phase-1 filtering, and all retained patterns are carried into the subsequent BPE merge step.

Finalization within phase 2. The merged state is finalized by running the full BPE merge loop up to the target vocabulary size:

- The merge loop uses `min_frequency=2`, the same numerical threshold used when saving shard states.
- Setting `min_frequency=2` at finalization preserves the behavior of the reported tokenizers.

The resulting tokenizer is saved as a Hugging Face-compatible `tokenizer.json`.

Frequency metadata. The merge routine records token-frequency metadata derived from the merged filtered state, including a token-count sidecar alongside the tokenizer artifact; the vocabulary-utilization diagnostics in Appendix D.5.3 use these counts to compute dead-merge and active-utilization statistics.

Vocabulary sizes and alphabet overhead. The core sweep used for the paper's 9 significance-tested conditions trains tokenizers at three vocabulary sizes: 16K, 32K, and 64K, with `101-lang` extensions at 128K and 256K. All rows include the same $S = 5$ special tokens, so the available merge slots are $V - S - 512$ for UBE and $V - S - 256$ for BBPE; Table B.1 reports the resulting accounting.

Table B.1: Merge-slot accounting by vocabulary size. All rows reserve the same five special tokens; UBE spends an additional 256 slots on its disjoint UTF-16 alphabet relative to BBPE.

Vocab size	UBE merges	BBPE merges	UBE overhead
16K	15,483	15,739	1.6%
32K	31,483	31,739	0.8%
64K	63,483	63,739	0.4%
128K	127,483	127,739	0.2%
256K	255,483	255,739	0.1%

Reproducibility. Our preparation scripts store each language's text as Arrow shards, reserve one terminal shard per language for held-out evaluation, and hold the raw-data manifest fixed across tokenizer families. The frozen 101-language mC4 [Xue et al., 2021] mirror therefore contains 2,273 train shards and 101 held-out evaluation shards overall. As discussed in the main text, `101-lang/32K` is the cleanest downstream attribution setting because UBE and BBPE see nearly identical raw-text coverage there; the lower-coverage settings are retained as fixed-compute stress tests.

Merge-order tie-breaking. The BPE merge loop is deterministic given the merged word-count table, but hash-map iteration order can change how equal-frequency ties near frequency boundaries are resolved. A retrain without deterministic tie handling is therefore *frequency-exact* (all merged pair counts are identical across runs) but not necessarily *bit-for-bit reproducible*: a small number of merges near frequency boundaries may be reordered.

The tokenizer directories used in this work are fixed and define the canonical artifacts for all reported experiments. For deterministic reruns of tie-prone training stages, our training pipeline can apply stable secondary keys to equal-frequency alphabet and post-merge-pruning ties.

As a sensitivity check, we retrained the `6-lang/32K` UBE tokenizer with this deterministic-tie mode. The retrained tokenizer keeps the same 32K vocabulary size and five special-token IDs and round-trips multilingual samples without `<unk>` emission. Table B.2 compares the original tokenizer to this deterministic-tie variant on the same intrinsic datasets. The differences are at most 0.0005 in Gini and 0.0028 in mean premium, so the paper's intrinsic conclusions are not driven by hash-map tie order in this stress setting. We treat this as a targeted sensitivity check rather than as a full deterministic retrain sweep across all vocabulary sizes.

B.2 Language-model training and evaluation configuration

The primary two-scale LM matrix spans the 182M/32K coverage matrix and the Chinchilla-style [Hoffmann et al., 2022] 1.3B/32K scale check. The reported core LM suite contains 16

Table B.2: Deterministic tie-break sensitivity check for UBE `6-lang/32K`. "Det." is the tokenizer trained with deterministic secondary ordering for equal-frequency merge candidates. Δ columns report Det. minus the original UBE tokenizer.

Dataset	Original Gini	Det. Gini	ΔGini	Δ mean premium
FLORES-200	0.2522	0.2517	-0.00050	-0.0028
UDHR	0.2473	0.2471	-0.00025	+0.0003
SIB-200	0.2600	0.2595	-0.00050	-0.0025
MGSM	0.3120	0.3120	+0.00002	-0.0004

runs: this primary matrix plus a separate 329M/128K composition check for S-UBE and S-BBPE. We additionally report 12 appendix LM runs: 6 230M/64K vocabulary variants, 2 182M/32K SuperBPE composition rows, and 4 extra random-seed repeats for the matched `101-lang/32K` setting. All runs keep the LM corpus fixed to the 101-language mC4 release; runs differ in tokenizer coverage, vocabulary size, model scale, or seed. Within each setting, architecture, optimizer, and token-step budget are matched while the tokenizer configuration changes; raw-text exposure can therefore differ in the lower-coverage fixed-compute stress rows. The LM dataloader applies no language-level temperature sampling or resampling: it deterministically shuffles a fixed, capped set of per-language shards, so the retained shard volumes determine the language mixture. Table B.3 summarizes the reported core suite, and Table B.4 reports the 32K token-budget estimates per configuration.

Transformer architecture. All LM runs use the same decoder-only LitGPT/LLaMA-style architecture. The 182M, 230M, and 329M runs use 14L/12H/768d (head dimension 64, MLP 3072); the 230M and 329M variants differ from the 182M/32K setting only by the 64K and 128K vocabularies, respectively. The 1.3B runs use 24L/16H/2048d (head dimension 128, MLP 5504). Blocks use pre-norm `RMSNorm` ($\epsilon = 10^{-5}$), sequential residuals (`parallel_residual=false`), standard causal multi-head attention with `n_query_groups=n_head` (not grouped-query or multi-query attention), no query-key normalization, no sliding-window attention, and no attention/MLP/projection biases. Positional encoding is RoPE only, applied to all query/key head dimensions (`rotary_percentage=1.0`, `rope_base=10000`, `rope_condense_ratio=1`), with no learned absolute position embeddings. The feed-forward sublayer is a LLaMA gated SiLU MLP, $\mathrm{SiLU}(xW_1) \odot xW_2$, followed by an output projection. Token embeddings and the LM head are untied; the vocabulary is read from each tokenizer artifact, padded to a multiple of 512, and uses the same five special tokens listed in Appendix A.

Table B.3: Reported core LM suite on 8×A100 80 GB GPUs. "Batch recipe" is per-device microbatch × gradient-accumulation steps; all rows use sequence length 2048. The 182M and 1.3B rows form the primary two-scale UBE/BBPE matrix; the 329M row is a separate 128K composition check for S-UBE and S-BBPE.

Setting	Runs	Layers/heads/width	Params	Batch recipe	Global tok/step	Role
182M	6	14/12/768	182M	32×1	524,288	full coverage matrix (200K steps, 105B tok)
329M	4	14/12/768	329M	8×4	524,288	128K composition check (200K steps, 105B tok)
1.3B	6	24/16/2048	1.35B actual	8×1	131,072	scale verification (205K steps, ∼26.9B tok)

Thus, the 1.3B row processes 8 GPUs × 8 sequences per device × 1 accumulation step × 2048 tokens = 131,072 global tokens per optimizer step. This row is a matched UBE/BBPE scale-verification experiment rather than the primary attribution setting.

Optimizer and training details. All LM runs use the same LitGPT-based trainer with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$), weight decay 0.1, and gradient clipping at $\|g\| \leq 1.0$. Training uses a linear learning-rate (LR) warmup over 4,000 steps followed by cosine decay to 10% of the peak learning rate. The 182M, 230M, and 329M settings use peak LR 6×10^{-4} with min LR 6×10^{-5}; the 1.3B setting uses peak LR 3×10^{-4} with min LR 3×10^{-5}. Sequence length is 2048 throughout. Mixed-precision training is bf16-mixed with `torch.compile`. All single-seed LM runs use the same seed; the multi-seed stability study (Appendix F.2) adds two independent seeds. During training,

Table B.4: Estimated 32K LM token accounting under the shared 101-language mC4 release. Train counts use one-shard-per-language byte-weighted extrapolation; evaluation counts enumerate held-out terminal shards. Steps/epoch are train-token totals divided by 524,288 global tokens/step, and fixed 200K steps correspond to 104.9B training tokens.

Tokenizer coverage	UBE train	BBPE train	UBE eval	BBPE eval	Steps/epoch	200K coverage
`En-only` on 101-lang mC4	∼166.9B	∼184.0B	∼7.2B	∼7.5B	318,320 / 350,991	62.8% / 57.0%
`6-lang` on 101-lang mC4	∼119.0B	∼120.0B	∼6.9B	∼7.0B	227,033 / 228,967	88.1% / 87.3%
`101-lang` on 101-lang mC4	∼107.8B	∼108.1B	∼5.3B	∼5.3B	205,695 / 206,142	97.2% / 97.0%

Entries with two values are UBE/BBPE. The 64K LM training runs (`En-only`/`6-lang`/`101-lang`) and 128K LM training runs (`101-lang` only) in Tables F.3 and F.4 share the same 101-language mC4 corpus and 200K-step budget. Their token totals follow the same one-shard-per-language methodology and trend slightly downward at larger vocabularies as more frequent patterns absorb into multibyte merges (cf. the `All avg` column of Table D.3).

a lightweight held-out perplexity check runs every $5{,}000$ steps over 70 iterations on the terminal evaluation shard. Dropout is disabled. Optimizer states are reset per run; no checkpoint resumption is used across tokenizer conditions.

Evaluation details. Held-out predictive loss is evaluated as per-language BPB on deterministic terminal-shard splits across all 101 mC4 language directories, with 1,600 context blocks per language to equalize representation. Blocks follow a seeded document order and each language contributes a fixed number of tokens, so the scored text varies with the tokenizer and the seed; Table E.8 instead scores a common held-out text. Let L_{tok} sum natural-log negative log-likelihood over all predicted targets and L_{text} sum it only over targets with positive raw-text byte counts, excluding EOS and other zero-byte targets. With N_{tok} predicted targets and N_{byte} raw UTF-8 bytes, the token-level perplexity is $\mathrm{PPL} = \exp(L_{\mathrm{tok}}/N_{\mathrm{tok}})$ and $\mathrm{BPB} = L_{\mathrm{text}}/(N_{\mathrm{byte}} \ln 2)$. Thus, token-level perplexity is not obtained by converting BPB alone. The BPB denominator uses the raw byte count of the underlying text for both UBE and BBPE, rather than a tokenizer-dependent encoded-byte count. Downstream evaluation uses the test splits for XNLI [Conneau et al., 2018] (15 languages), XCOPA [Ponti et al., 2020] (11), Belebele [Bandarkar et al., 2024] (122), ARC-Easy/ARC-Challenge [Clark et al., 2018], and LAMBADA [Paperno et al., 2016]; HellaSwag [Zellers et al., 2019] and PIQA [Bisk et al., 2020] use their validation splits.

B.3 Corpus and token-count accounting

The primary corpus uses tiered per-language targets measured in stored Arrow data: approximately 100 GiB for English, 30 GiB each for Korean, Chinese, Japanese, Hindi, and Thai, and 1 GiB for each of the remaining 95 languages, subject to data availability and shard granularity. As in other multilingual tokenizers, these targets depart from the natural size distribution of the source data: prior work resamples languages so that smaller ones are well represented in the vocabulary [Conneau and Lample, 2019, Xue et al., 2021, NLLB Team, 2024], and one multilingual BPE tokenizer study fixes English at 30% of its tokenizer data [Abagyan et al., 2026]. Our tiers give English the largest share, give five languages whose primary scripts use 3-byte UTF-8 characters a data-rich partition, and keep the other 95 languages in tokenizer training at a smaller fixed size. Because the targets are byte counts, the content of a partition also depends on its *byte premium*, which measures UTF-8 byte-length ratios between content-matched texts [Arnett et al., 2024]; the encoding floor is a scalar-level, pre-merge analogue of this corpus-level measure. Table B.5 summarizes the three tokenizer-training subsets, which share the same underlying shards and differ only in language coverage. Within each configuration–vocabulary comparison, tokenizer families use identical raw-text shards, allowing us to compare byte representations under a fixed, non-uniform language mixture; Appendix E.7 repeats the comparison on a 50 MiB/language prefix. The LM corpus is held fixed to the full 101-language release across all primary and appendix LM runs.

The `6-lang` set comprises {en, zh, ko, ja, hi, th}, chosen as a compact mixed-script stress set with Latin text (en), CJK scripts for zh/ko/ja, and Indic/Thai scripts for hi/th. It does not include a training language written in a 2-byte UTF-8 script such as Arabic, Hebrew, Greek, or Cyrillic. The half-data ablation in Appendix D.4 uses approximately half of the 346 GiB `101-lang` corpus (∼173 GiB), obtained by per-language uniform shard downsampling at fraction 0.5.

Table B.5: Tokenizer-training corpus size by configuration. Rows aggregate the language subset used to train each tokenizer; the LM corpus always uses the full 101-language mC4 release. Evaluation columns enumerate held-out terminal shards, document counts are byte-weighted one-shard estimates, and raw byte totals are exact Arrow file sizes.

Config	Langs	Train			Eval	
		Shards	GiB	Docs	Shards	Docs
`En-only`	1	118	101	32.1M	1	0.27M
`6-lang`	6	1,330	253	62.3M	6	0.46M
`101-lang`	101	2,273	346	89.3M	101	4.36M

Tokenizer-dependent token counts. Token counts in Table B.4 are estimated by one-shard-per-language byte-weighted sampling (middle interior train shard only, excluding the last potentially smaller shard). Evaluation shards are fully enumerated. Both tokenizers sample the same shard per language (fixed seed) for fair comparison. The table reports the 32K primary attribution scale and folds train-token totals, evaluation-token totals, steps per epoch, and 200K-step coverage into one accounting table. The 64K LM training runs cover all three coverage configs and the 128K LM training runs cover `101-lang` only (Tables F.3 and F.4); they share the same 101-language mC4 release, the same one-shard-per-language sampling, and the same fixed 200K-step budget. Their per-language token-per-byte rates decrease with vocabulary size as more frequent patterns are absorbed into multibyte merges (the `All avg` column of Table D.3 illustrates the same intrinsic scaling pattern under `101-lang`).

All six tokenizers are applied to the same 101-language mC4 corpus. `En-only` tokenizers inflate non-English text to more tokens because low-coverage spans remain closer to raw-byte decomposition, so the same raw bytes yield more training tokens per epoch. The 200K fixed-step budget keeps every 32K run in Table B.4 under one epoch. The `101-lang` comparison is therefore the closest tokenizer-isolation setting, whereas `En-only` and `6-lang` should be read as fixed-compute stress tests that intentionally include raw-text exposure differences.

C Intrinsic examples and checks

C.1 Representative language examples

Table C.1 lists representative languages outside the 6-language tokenizer training set; BBPE remains close to the encoding floor for Amharic, Lao, and Santali, while its merges already shorten Telugu, Kannada, and Tibetan. Unlike the core `101-lang/32K` examples in the main text, this narrower `6-lang/32K` setup is intentionally a stress regime. Table C.2 extends the same six languages to the 16K and 64K `6-lang` vocabularies; the `6-lang` setting is not trained at 128K–256K (only `101-lang` is extended that far, see Appendix D.1). Table C.3 extends the main-text 101-lang representative comparison (Table 3) across all five vocabulary sizes (16K–256K).

Table C.1: Per-language UBE impact on FLORES-200 in the `6-lang/32K` stress setting (mean tokens per sentence). The listed non-English languages are outside tokenizer training; UBE cuts token counts by 6–35%, equivalent to 6–53% more fixed-budget content. The 16K and 64K results for the same languages appear in Table C.2.

Language	Region	Speakers	UBE	BBPE	Δ%	Script
Amharic	Ethiopia	∼57M	**137.539**	202.775	-32.2%	Ethiopic
Lao	Laos	∼30M	**233.445**	342.100	-31.8%	Lao
Santali	India	∼8M	**222.707**	341.287	-34.7%	Ol Chiki
Telugu	India	∼82M	**220.807**	237.530	-7.0%	Telugu
Kannada	India	∼44M	**237.506**	251.956	-5.7%	Kannada
Tibetan	China/Nepal	∼6M	**247.286**	264.246	-6.4%	Tibetan
English	(ref.)	—	**30.169**	30.249	-0.3%	Latin

Speaker-count note. Approximate Ethnologue 27th edition figures may include macrolanguage clusters and L2 speakers; the Lao count includes closely related Isan varieties, and the Tibetan count covers the Tibetic language cluster rather than Standard Tibetan alone [Eberhard et al., 2024].

Table C.2: Per-language UBE impact on FLORES-200 across the `6-lang` vocabulary sweep (16K/32K/64K; mean tokens per sentence). The same six representative non-English languages as Table C.1 are shown. $\Delta\%$ is computed as $(\text{UBE} - \text{BBPE})/\text{BBPE}$.

Language	Script	16K			32K			64K		
		UBE	BBPE	$\Delta\%$	UBE	BBPE	$\Delta\%$	UBE	BBPE	$\Delta\%$
Amharic	Ethiopic	**140.388**	219.321	-36.0%	**137.539**	202.775	-32.2%	**134.837**	163.896	-17.7%
Lao	Lao	**234.068**	343.176	-31.8%	**233.445**	342.100	-31.8%	**231.798**	256.355	-9.6%
Santali	Ol Chiki	**230.685**	363.850	-36.6%	**222.707**	341.287	-34.7%	**213.132**	341.274	-37.5%
Telugu	Telugu	**224.362**	263.163	-14.7%	**220.807**	237.530	-7.0%	**181.593**	184.658	-1.7%
Kannada	Kannada	**238.641**	354.337	-32.7%	**237.506**	251.956	-5.7%	**219.524**	220.944	-0.6%
Tibetan	Tibetan	**287.103**	336.849	-14.8%	**247.286**	264.246	-6.4%	**247.167**	253.668	-2.6%
English	Latin	**33.451**	33.629	-0.5%	**30.169**	30.249	-0.3%	**28.035**	28.078	-0.2%

Table C.3: Representative per-language UBE impact on FLORES-200 across `101-lang` vocabulary sizes (average tokens per sentence). Extends Table 3.

Language	Script	Vocab	UBE	BBPE	$\Delta\%$
Santali	Ol Chiki	16K	**237.326**	341.312	-30.467%
		32K	**222.944**	341.287	-34.676%
		64K	**212.692**	341.271	-37.677%
		128K	**211.712**	251.125	-15.695%
		256K	**207.379**	228.583	-9.276%
Central Atlas Tamazight	Tifinagh	16K	**192.112**	269.200	-28.636%
		32K	**183.305**	269.166	-31.899%
		64K	**182.243**	269.123	-32.282%
		128K	**180.664**	269.102	-32.864%
		256K	**180.648**	203.206	-11.101%
Tibetan	Tibetan	16K	**287.065**	432.437	-33.617%
		32K	**286.737**	336.804	-14.865%
		64K	**247.171**	264.208	-6.449%
		128K	230.166	**214.252**	+7.428%
		256K	163.501	**161.215**	+1.418%
Arabic	Arabic	16K	**68.952**	69.872	-1.317%
		32K	**60.407**	60.600	-0.318%
		64K	**52.653**	52.753	-0.189%
		128K	**46.245**	46.296	-0.111%
		256K	**40.890**	40.905	-0.036%
English	Latin	16K	**34.671**	34.863	-0.550%
		32K	**31.068**	31.160	-0.295%
		64K	**28.695**	28.716	-0.076%
		128K	**27.158**	27.174	-0.058%
		256K	**26.271**	26.272	-0.004%

C.2 Naturally occurring mixed-script documents

The main-text mixed-script claim is measured on 400 naturally occurring mC4 documents drawn uniformly from 8 languages (`ja`, `ko`, `zh`, `hi`, `th`, `ar`, `ru`, `bn`; 50 documents each), tokenized under the `6-lang/32K` tokenizer family. UBE uses fewer tokens than BBPE in 7 of 8 languages; Bengali is the only language that regresses (+0.4%). Aggregated over all 400 documents, UBE uses 0.2% fewer tokens than BBPE, while BBPE16 is 6.6% longer because Latin runs in the same documents inherit the uniform 2-byte floor. We report this slice at the matched `6-lang/32K` setting because it uses the same compact tokenizer as the 6-language stress examples in Table C.1; the qualitative BBPE16/UBE Latin-floor pattern is consistent across vocabulary sizes (cf. `EN avg` and `Latin avg` columns of Table D.3 from 16K to 256K). The per-document inputs are selected deterministically by our sampling script from the same public mC4 subset used for tokenizer training, so the selection depends only on public inputs.

C.3 Contamination checks

Following prior overlap-based contamination audits such as Arnett et al. [2025], we compare the mC4 scan corpus (89.47M documents across 101 languages) against the evaluation sentences of each benchmark using two surface-form checks: exact document–sentence equality and a 50-character prefix-substring scan. The 89.47M figure is the number of mC4 documents loaded by this scan, not the full tokenizer-training document estimate in Table B.5. Table C.4 reports the overlap counts.

Table C.4: Surface-form contamination audit against the mC4 scan corpus. Prefix overlaps use the 50-character prefix-substring check described in the text.

Evaluation suite	Eval sentences	Exact overlaps	Prefix overlaps
FLORES-200	206,448	0	53
UDHR	35,870	1	7,102
SIB-200	41,820	0	10
MGSM	2,750	0	0

The prefix-substring scan covers only sentences with at least 50 characters: 201,686 FLORES-200, 22,133 UDHR, 40,771 SIB-200, and 2,717 MGSM sentences. The exact-match scan found no FLORES-200, SIB-200, or MGSM overlaps, and one UDHR overlap. Prefix overlaps are rare for FLORES-200 and SIB-200 and absent for MGSM, while UDHR has many prefix overlaps, consistent with its status as a short, widely reproduced public UN text. We therefore treat the UDHR result as a robustness stress test rather than as evidence from a fully held-out corpus.

D Intrinsic scaling and robustness diagnostics

D.1 Training config ablation

Table D.1: FLORES-200 language-level outcome summary across configurations (204 languages). Columns show languages where UBE produces fewer tokens (wins), ties, and more tokens (losses), plus the corresponding Gini change. The 128K and 256K rows are only run for `101-lang`.

Config	Vocab	UBE wins/ties/losses	ΔGini
`En-only`	16K	37 / 19 / 148	-13.9%
`En-only`	32K	41 / 23 / 140	-11.9%
`En-only`	64K	37 / 82 / 85	-7.3%
`6-lang`	16K	202 / 0 / 2	-8.7%
`6-lang`	32K	191 / 0 / 13	-7.2%
`6-lang`	64K	169 / 24 / 11	-6.2%
`101-lang`	16K	196 / 0 / 8	-12.1%
`101-lang`	32K	189 / 0 / 15	-9.4%
`101-lang`	64K	187 / 0 / 17	-8.2%
`101-lang`	128K	174 / 5 / 25	-5.2%
`101-lang`	256K	150 / 13 / 41	-1.9%

Across the 9 core tokenizer–vocabulary configurations (16K–64K), UBE reduces token-premium variance by 24%–49% and Gini coefficient by 6%–14%. For the primary dispersion statistic, a paired language bootstrap over non-English token premiums (10K resamples) places ΔGini (UBE − BBPE) below zero in every core setting; representative 95% CIs are [-0.0616, -0.0323] for `En-only/16K`, [-0.0358, -0.0047] for the primary `101-lang/32K` setting, and [-0.0299, -0.0044] for `6-lang/64K`. Separately, Wilcoxon signed-rank tests over language-level mean token counts are significant in 8/9 comparisons at $p < 0.01$; the same 8/9 token-count decision is unchanged under Holm correction at $\alpha = 0.05$ and Benjamini–Hochberg false discovery rate (FDR) correction at $q = 0.05$ over the nine prespecified core comparisons. The lone token-count exception is the near-tie `En-only/64K` setting (max $p = 0.32$). Theil-T and Theil-L also decrease in all nine settings, so the disparity ordering is not specific to Gini. The largest qualitative examples are not the typical case: outside the most floor-bound scripts, many per-language token-count differences are small, while the aggregate distribution still shifts consistently toward lower disparity.

Only the `101-lang` setting was extended to 256K, so the corresponding standalone and composition sweeps are collected in Tables D.3 (Appendix D.2) and D.4 (Appendix D.3).

D.2 Gini coefficient and absolute token counts

We report two fixed diagnostic groups of 3-byte BMP languages alongside overall results for all 204 languages. CJK comprises Chinese (Simplified and Traditional), Cantonese, Japanese, and Korean; the low-coverage BMP (LC-BMP) languages are the Tibetan-, Myanmar-, Ol Chiki-, and Tifinagh-script languages (Dzongkha, Tibetan, Burmese, Shan, Santali, Tamasheq, and Central Atlas Tamazight). In both groups, a core character costs three UTF-8 bytes, which UBE lowers to two base symbols. The groups differ in the other factor of a byte premium, the number of characters needed for the same content [Arnett et al., 2024]: relative to English, parallel FLORES-200 text uses 0.31–0.51 times as many characters in CJK but 0.89–1.42 times as many in the LC-BMP languages [CANINE premiums in Appendix C of Petrov et al., 2023], and the LC-BMP languages hold seven of the eight highest token premiums under GPT-4's cl100k_base [Petrov et al., 2023]. Training coverage differs as well: the Korean, Chinese, and Japanese partitions each hold about 30 GiB of the primary corpus, whereas the LC-BMP group has only the 1 GiB Burmese partition and no dedicated partition for its other six languages. The split therefore separates scripts with large dedicated training partitions from low-coverage scripts, which have little or no dedicated training data in the primary corpus. Both groups are unchanged across vocabulary sizes and composition settings.

Table D.2: FLORES-200 Gini for UBE and BBPE under the primary `101-lang` training regime. Table D.3 gives the token-count breakdown.

Vocab	UBE Gini	BBPE Gini	ΔGini	UBE wins
16K	**0.1927**	0.2194	-12.1%	196/204
32K	**0.1979**	0.2183	-9.4%	189/204
64K	**0.2043**	0.2225	-8.2%	187/204
128K	**0.2140**	0.2258	-5.2%	174/204
256K	**0.2204**	0.2246	-1.9%	150/204

Table D.3: Gini coefficient and groupwise mean tokens per sentence on FLORES-200 (`101-lang`; 16K–256K). Latin avg covers Latin-script languages; the CJK and LC-BMP groups are defined in Appendix D.2. †BBPE16 lowers Gini via a uniform 2-byte floor that raises Latin token counts; SuperBPE compositions are reported separately.

Tokenizer	EN avg	Latin avg	CJK	LC-BMP	All avg	Gini
UBE 16K	**34.671**	**62.163**	**48.052**	**228.410**	**76.410**	**0.19274**
UBE 32K	**31.068**	**56.627**	**42.418**	**219.709**	**69.857**	**0.19790**
UBE 64K	**28.695**	**52.500**	37.438	**204.090**	**64.448**	**0.20428**
UBE 128K	**27.158**	**48.722**	33.596	**189.992**	**59.649**	**0.21401**
UBE 256K	**26.271**	**45.097**	30.498	**167.665**	**55.105**	**0.22039**
BBPE 16K	34.863	62.373	48.521	308.522	79.408	0.21938
BBPE 32K	31.160	56.743	42.444	275.681	71.889	0.21833
BBPE 64K	28.716	52.540	**37.237**	250.389	66.056	0.22254
BBPE 128K	27.174	48.751	**33.353**	217.459	60.594	0.22576
BBPE 256K	26.272	45.100	**30.259**	176.498	55.389	0.22456
BBPE16 16K	47.935	90.432	51.045	228.970	95.683	0.14588†
BBPE16 32K	41.317	82.137	45.083	222.753	87.170	0.15982†
BBPE16 64K	37.054	74.313	39.576	208.599	79.224	0.17218†
BBPE16 128K	34.471	67.506	35.364	201.591	72.668	0.18906†
BBPE16 256K	32.929	61.839	31.876	174.085	66.468	0.19920†

D.2.1 Why Gini rises with vocabulary size

Figure D.1 visualizes the mechanism behind the Gini increase observed in Table D.3. Script families compress at vastly different rates. Across the 16K–256K vocabulary sweep, English token counts

fall by 24.2%, whereas Hindi (Devanagari) token counts fall by only 5.6%. Because premiums are ratios to English, Hindi's premium *rises* from 2.57 to 3.20 (+0.63) even though its absolute count decreases slightly. Meanwhile, languages that can still form long merges (Arabic: 40.7% reduction, premium -0.43; Korean: 37.6%, -0.23) converge toward English. The net effect is a widening premium distribution—75 of the 203 non-English languages see their premium increase—that drives Gini upward. This pattern is expected for BPE on a multilingual corpus when merge productivity is script-dependent: some script families absorb more productive merges per added vocabulary slot than others. UBE keeps Gini below BBPE's at every size but does not remove this vocabulary-scaling effect.

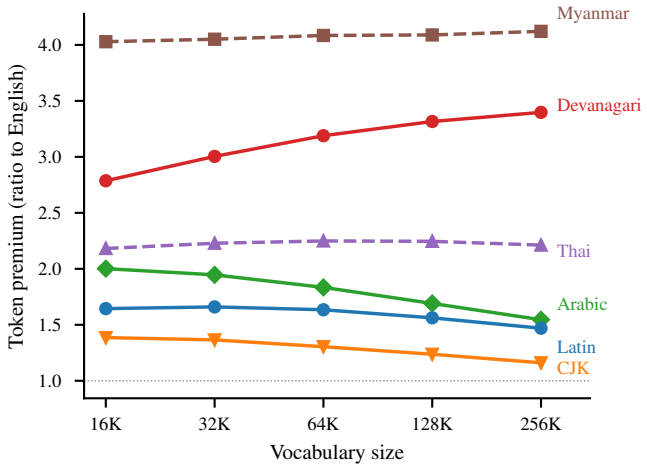

Figure D.1: Token premium (ratio to English) vs. vocabulary size for six script families under UBE (`101-lang`).

D.2.2 First-order token-cost intuition

Complementing the empirical premium-scaling decomposition in Appendix D.2.1, a simple first-order cost sketch provides intuition for (i) the dataset ordering in Table D.12 (UDHR > FLORES-200 > MGSM at 32K–256K) and (ii) the monotonic attenuation of UBE's Gini reduction—that is, ΔGini approaching zero—as vocabulary grows (Table D.2). We present it as an *intuition*, not as a quantitative prediction: UBE changes the merge-slot budget and the observed vocabulary utilization (Appendix D.5.3), so the constant-coverage derivation below is a first-order sketch rather than a parameter-free forecast.

Setup. For a script s and substrate $E \in \{\text{UBE}, \text{BBPE}\}$, let $f_s^{(E)}$ denote the corresponding main-text encoding-floor value. Under BBPE, $f_s^{(\text{BBPE})} = |\text{utf8}(c)|$ for c drawn from s, taking values in $\{1, 2, 3, 4\}$. Under UBE, the routing rule sets $f_s^{(\text{UBE})} = f_s^{(\text{BBPE})}$ for $f_s \leq 2$, $f_s^{(\text{UBE})} = 2$ for $f_s = 3$, and $f_s^{(\text{UBE})} = 4$ for $f_s = 4$. Let $c_s^{(E)} \in [0, 1]$ denote the *character-level merge coverage* under substrate E: the probability that a random character from script s is absorbed by a learned BPE token whose per-character cost is below $f_s^{(E)}$. Let $m_s^{(E)}$ denote the expected token count per character among covered characters.

Model. The expected token cost per character under substrate E is

$$T_s^{(E)} \approx c_s^{(E)} m_s^{(E)} + \left(1 - c_s^{(E)}\right) f_s^{(E)}.$$

This is a first-order approximation that treats coverage as a scalar and ignores correlations between adjacent characters.

UBE effect under a constant-coverage assumption. UBE's alphabet, merge-slot budget, and merge-utilization structure all differ from BBPE's (Appendix D.5.3), so $c_s^{(E)}$ and $m_s^{(E)}$ are genuinely

substrate-dependent. If we simplify by assuming $c_s^{(\mathrm{UBE})} \approx c_s^{(\mathrm{BBPE})} =: c_s$ and $m_s^{(\mathrm{UBE})} \approx m_s^{(\mathrm{BBPE})} =: m_s$, the UBE–BBPE difference collapses, for 3-byte BMP scripts, to

$$T_s^{(\mathrm{UBE})} - T_s^{(\mathrm{BBPE})} \;\approx\; (1 - c_s)\left(f_s^{(\mathrm{UBE})} - f_s^{(\mathrm{BBPE})}\right) \;=\; -(1 - c_s).$$

For ASCII and 2-byte UTF-8 scripts, $f_s^{(\mathrm{UBE})} = f_s^{(\mathrm{BBPE})}$, so the first-order saving is zero. For 3-byte BMP scripts, by contrast, the floor change—and hence the first-order saving above—is scaled by the uncovered fraction $1 - c_s$.

Qualitative dataset ordering. Let $\bar{u}_D = \mathbb{E}_{s \sim D,\, f_s = 3}\left[1 - c_s^{(\mathrm{BBPE})}\right]$ denote the mean uncovered fraction of 3-byte characters in dataset D under BBPE. We do not measure $\bar{u}_D$; the following large, moderate, and near-zero labels follow from dataset coverage breadth alone:

- UDHR (387 corpus entries, many with no mC4 training data): $\bar{u}_D$ is expected to be large. Observed ΔGini at 32K: -11.1%.
- FLORES-200 (204 languages, mixed coverage): $\bar{u}_D$ is expected to be moderate. Observed: -9.4%.
- MGSM (11 languages, all of which appear in the 101-language mC4 corpus): $\bar{u}_D$ is expected to be near zero. Observed: +0.1%.

The observed ordering is consistent with this first-order sketch. We rank $\bar{u}_D$ from coverage breadth rather than measuring it directly; per-dataset measurement against the trained BBPE tokenizer is left to future work.

Decay with vocabulary size. Larger vocabularies increase $c_s^{(\mathrm{BBPE})}$ for frequent patterns, shrinking $(1 - c_s^{(\mathrm{BBPE})})$ and therefore the first-order UBE saving. This is consistent with ΔGini monotonically approaching zero from -12.1% at 16K to -1.9% at 256K (Table D.2) and the per-script premium-scaling decomposition in Appendix D.2.1.

Beyond the floor-only saving. Under the constant-coverage assumption above, replacing a 3-byte floor with 2 bytes yields at most a $1/3$ premium reduction. Empirically, Santali at `101-lang/64K` has a UBE/BBPE token-count ratio below the corresponding $2/3$ floor-only limit (Table D.5), exceeding this floor-only saving. UBE's UTF-16 path therefore enables BPE merges that absorb content beyond what BBPE's UTF-8 path captures, so the constant-coverage assumption underestimates the realized saving.

D.3 SuperBPE composition results

Complementing the standalone UBE/BBPE and BBPE16 results in Table D.3, Table D.4 reports the full S-UBE/S-BBPE composition sweep. The table adds two details to the main-text summary: among the SuperBPE composition rows, S-UBE has lower Gini than S-BBPE at every vocabulary size, while plain UBE keeps fewer total FLORES-200 tokens than S-UBE at 16K–64K; and the bounded, sign-changing interaction term supports a qualitative complementarity claim rather than exact additivity.

Implementation of S-UBE and S-BBPE. We implement S-UBE and S-BBPE as SuperBPE-style *extension* runs, not as separately designed tokenizers. Each reported target vocabulary size $V \in \{16\mathrm{K}, 32\mathrm{K}, 64\mathrm{K}, 128\mathrm{K}, 256\mathrm{K}\}$ starts from an already trained Stage 1 tokenizer of the same substrate at vocabulary size $V/2$ (UBE for S-UBE, BBPE for S-BBPE), loads its `merges.txt` as ordered seed merges, and then continues BPE training until the total vocabulary reaches V. The Stage 1 seed tokenizers are intermediate artifacts only; all reported S-UBE/S-BBPE result rows use the 16K–256K target vocabularies. This mirrors the two-stage structure of SuperBPE: Stage 1 learns an ordinary whitespace-splitting BPE tokenizer over the chosen substrate, and Stage 2 reuses those merges while relaxing the pre-tokenizer so that cross-word merge candidates can be selected.

The Stage 2 implementation is shared by the two composed variants. It first regenerates shard-local count states using a permissive regex pre-tokenizer rather than the normal GPT-2 whitespace-splitting regex; these states cannot be reused from UBE/BBPE training because they contain multi-word chunks that ordinary pre-tokenization never exposes. Shard-local pre-tokenized chunks occurring

fewer than five times are discarded before aggregation; new merges use a minimum frequency of two. It then merges the shard states, initializes the BPE trainer with the substrate-specific base alphabet and the Stage 1 seed merges, replays the seed merges in their original order, rebuilds the pair-count priority queue from the permissive-regex corpus state, and resumes the standard greedy BPE merge loop. The only substrate-specific branch is the byte representation used before BPE: S-UBE uses UBE's UniversalByteLevel alphabet, pre-tokenizer, and decoder, so routed UTF-16 PUA symbols participate in the same SuperBPE Stage 2 merge loop; S-BBPE uses the standard UTF-8 byte-level alphabet, pre-tokenizer, and decoder. In both cases we retain SuperBPE's colon-space block and four-word cap (at most four whitespace-separated words per merged token), which avoids unbounded multiword tokens; the reported tokenizer sidecars record the base type, seed-merge count, selected shard-state count, target vocabulary, and these SuperBPE constraints.

Table D.4: SuperBPE composition results on FLORES-200 (204 evaluation languages; tokenizers trained under the `101-lang` mC4 regime; 16K–256K vocab). Format matches Table D.3.

Tokenizer	EN avg	Latin avg	CJK	LC-BMP	All avg	Gini
SuperBPE UBE 16K	**38.099**	**66.534**	**49.281**	**221.587**	**80.676**	**0.18269**
SuperBPE UBE 32K	**32.973**	**60.992**	**43.496**	**192.162**	**72.137**	**0.17248**
SuperBPE UBE 64K	**29.050**	**55.491**	38.808	**178.255**	**64.957**	**0.16790**
SuperBPE UBE 128K	**26.314**	**51.359**	34.539	**172.071**	**59.474**	**0.16998**
SuperBPE UBE 256K	**24.344**	**47.451**	31.223	**159.784**	**54.257**	**0.17280**
SuperBPE BBPE 16K	38.308	66.840	49.655	273.239	82.853	0.19967
SuperBPE BBPE 32K	33.143	61.177	43.595	245.653	74.187	0.19217
SuperBPE BBPE 64K	29.133	55.586	**38.710**	231.689	66.871	0.18996
SuperBPE BBPE 128K	26.337	51.476	**34.349**	215.979	61.045	0.19042
SuperBPE BBPE 256K	24.357	47.479	**30.977**	187.800	55.220	0.18694

Interaction-term stability. The interaction term of the 2×2 factorial, S-UBE% $-$ UBE% $-$ S-BBPE% on signed Gini percentage changes relative to BBPE (negative terms are Gini reductions), is bounded but sign-changing across vocabulary sizes: +4.4% at 16K, +0.3% at 32K, -1.7% at 64K, -3.8% at 128K, and -4.4% at 256K (mean -1.1%). A positive value indicates that the composition falls short of the summed individual reductions (interference); a negative value indicates that the composition exceeds the sum (mild synergy). This pattern supports a qualitative complementarity interpretation, not a uniform near-additive law: small-vocabulary alphabet overhead and larger-vocabulary cross-word merge opportunities affect the sign of the interaction. At 64K, a bootstrap over 203 non-English languages (10K resamples) places the interaction at -1.7% with a 95% CI of [-3.2%, -0.1%], indicating that this negative interaction point is small but distinguishable from zero. We do not claim a uniform interaction sign across vocabulary sizes.

D.4 Data-to-vocabulary ratio ablation

Figure D.2 plots the Gini coefficient as a function of vocabulary size for full-data (346 GiB) and half-data ($\sim$173 GiB) tokenizers across all four families. Full and half curves nearly overlap at every vocabulary size for all families, with the median $|\Delta\text{Gini}|$ across 20 comparisons of just 0.0011 (max 0.0087 at S-BBPE 16K). This indicates that the Gini increase with vocabulary size observed for UBE and BBPE is not explained simply by insufficient tokenizer-training data; Appendix D.2.1 traces it to script families that compress at different rates.

D.5 Per-language distribution and mechanism diagnostics

The full 16K–256K numerical sweep appears in Tables C.3, D.2, and D.3, with complete per-language values in Appendix I.2. Table D.5 lists the top 10 and bottom 10 languages by UBE/BBPE token ratio under the representative `101-lang/64K` setting, using raw mean token counts. Figure D.3 shows the best and worst five languages across all 16K–256K vocabulary sizes. Most languages cluster near parity (ratio ≈ 1.0), with a left tail of substantial improvements concentrated in low-resource 3-byte scripts. Across vocabulary sizes, the regression tail stays narrow and concentrated in a small recurring set of languages—chiefly Georgian, Amharic, Shan, and Tibetan, with Mossi (`mos_Latn`) as the Latin-script language that regresses most consistently across 32K–256K—and peaks at +7.4%

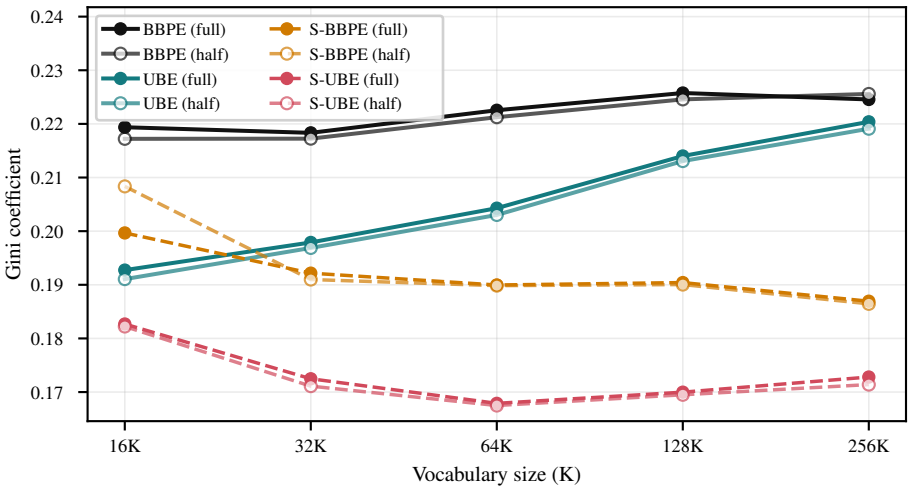

Figure D.2: Gini coefficient vs. vocabulary size for full-data (346 GiB) and half-data (∼173 GiB) tokenizer training (`101-lang`). Solid markers denote full data and open markers denote half data.

Table D.5: Best and worst 10 languages by UBE/BBPE token ratio on FLORES-200 (`101-lang`; UBE/BBPE: 64K, BBPE16: 128K). Ratio < 1 means UBE uses fewer tokens; ratio > 1 means UBE uses more. BBPE16 is included as a cross-substrate reference.

Language	UBE avg	BBPE avg	BBPE16 (128K) avg	UBE/BBPE
Top 10: UBE helps most				
Santali	**212.692**	341.271	222.620	0.6232
Central Atlas Tamazight	**182.243**	269.123	183.143	0.6772
Tamasheq (Tifinagh)	**185.609**	270.571	187.934	0.6860
Dzongkha	**265.685**	286.182	267.341	0.9284
Tibetan	**247.171**	264.208	247.411	0.9355
Kabyle	**62.457**	62.967	81.116	0.9919
Sicilian	**52.945**	53.291	75.957	0.9935
Lao	**87.657**	88.180	83.831	0.9941
Armenian	**64.792**	65.068	63.260	0.9957
Czech	**47.946**	48.132	53.918	0.9961
Bottom 10: UBE helps least (or regresses)				
Japanese	40.783	**40.630**	37.926	1.0037
Yoruba	70.871	**70.604**	77.817	1.0038
Chinese (Simplified)	35.015	**34.764**	33.126	1.0072
Chinese (Traditional)	39.077	**38.741**	36.869	1.0087
Cantonese	38.206	**37.856**	36.009	1.0092
Mossi	58.559	**57.896**	80.328	1.0115
Odia	135.328	**133.265**	121.959	1.0155
Georgian	71.468	**70.135**	64.738	1.0190
Amharic	61.581	**60.261**	55.687	1.0219
Shan	218.032	**204.210**	190.272	1.0677

for Tibetan at 128K. Dzongkha shares the Tibetan script but does not regress repeatedly; it improves at every vocabulary size except 128K.

D.5.1 Token composition and decomposition

Tables D.6 and D.7 illustrate the mechanism at the word level. For short Ol Chiki and Tifinagh words, BBPE produces exactly 3 tokens per character because it has learned virtually no merges for these scripts even at 64K vocabulary: each character's 3 UTF-8 bytes remain as separate base symbols. UBE starts at 2 base symbols per character via the UTF-16 path; for kemikal and the two

sentence fragments, BPE training has additionally learned cross-character byte merges that push the token count below the 2-per-character baseline. For a 24-character Ol Chiki sentence fragment, the combined floor and merge advantage yields 39 UBE tokens versus 63 BBPE tokens (38% reduction). Table D.8 extends the same word- and sentence-level examples across the full 16K–256K vocabulary sweep, and Table D.9 reports the corresponding per-script-family averages on FLORES-200.

Table D.6: Token composition for the scripts with the largest BBPE-to-UBE token-count gaps in the `101-lang`/64K setting. Per-script-family token counts across all 16K–256K vocabularies appear in Table D.9.

Script	Language	Codepoint	Floor (bytes/char)		Avg tokens/sent		Save
			UTF-8	UTF-16	UBE	BBPE	
Ol Chiki	Santali	`U+1C5E`	3	2	**212.692**	341.271	37.7%
Tifinagh	Central Atlas Tamazight	`U+2D30`	3	2	**182.243**	269.123	32.3%
Tibetan	Dzongkha	`U+0F42`	3	2	**265.685**	286.182	7.2%

Table D.7: Word- and sentence-level tokenization examples in the `101-lang`/64K setting. "Base" and "Tokens" count pre-merge symbols and final BPE tokens. "Word" gives a romanized form; "(*n* chars)" denotes sentence fragments. Table D.8 extends the same examples to all 16K–256K vocabularies.

Script	Word	Chars	UBE		BBPE		Save
			Base	Tokens	Base	Tokens	
Ol Chiki	ona	3	**6**	**6**	9	9	33%
Ol Chiki	kemikal	7	**14**	**12**	21	21	43%
Tifinagh	sin	3	**6**	**6**	9	9	33%
Tifinagh	acukula	7	**14**	**14**	21	21	33%
Ol Chiki	(24 chars)	24	**45**	**39**	66	63	38%
Tifinagh	(18 chars)	18	**34**	**31**	50	48	35%

Table D.8: Word- and sentence-level token counts across the 16K–256K vocabulary sweep (`101-lang`), using the same inputs as Table D.7.

Script	Word	Chars	16K		32K		64K		128K		256K	
			UBE	BBPE	UBE	BBPE	UBE	BBPE	UBE	BBPE	UBE	BBPE
Ol Chiki	ona	3	**6**	9	**6**	9	**6**	9	**6**	**6**	**6**	**6**
Ol Chiki	kemikal	7	**14**	21	**13**	21	**12**	21	**12**	14	**10**	14
Tifinagh	sin	3	**6**	9	**6**	9	**6**	9	**6**	9	**6**	**6**
Tifinagh	acukula	7	**14**	21	**14**	21	**14**	21	**14**	21	**14**	**14**
Ol Chiki	(24 chars)	24	**42**	63	**40**	63	**39**	63	**39**	45	**38**	42
Tifinagh	(18 chars)	18	**31**	48	**31**	48	**31**	48	**31**	48	**31**	34

Merge allocation and English token counts. Under UBE, one merge turns a 3-byte BMP character into a single token; BBPE needs two, the first producing a partial character [Land and Arnett, 2025]. Consistent with this, BPE spends fewer merges under UBE on 3- and 4-byte characters and on partial characters: 505–684 fewer in each multilingual setting (`6-lang` at 16K–64K and `101-lang` at 16K–256K; 10,563 vs. 11,211 at `101-lang/32K`). Despite the 256 extra base symbols, UBE gains mostly ASCII-only subwords (214–370 more; 17,118 vs. 16,805 at `101-lang/32K`) and some subwords with 2-byte characters (3–113 more). At `101-lang/32K`, UBE has 319 ASCII-only entries absent from BBPE, and BBPE learns 317 of them later in its frequency-ordered merge list [Hayase et al., 2024]: they appear in the `101-lang/64K` BBPE tokenizer, whose first 31,739 merges are exactly those of the 32K one. Of all uses of these 319 entries on FLORES-200, 99% fall in Latin-script languages. English shares this gain: UBE uses 93 fewer English tokens, 90 of them in ASCII-only pieces, whereas curly quotes and dashes are single tokens under both tokenizers.

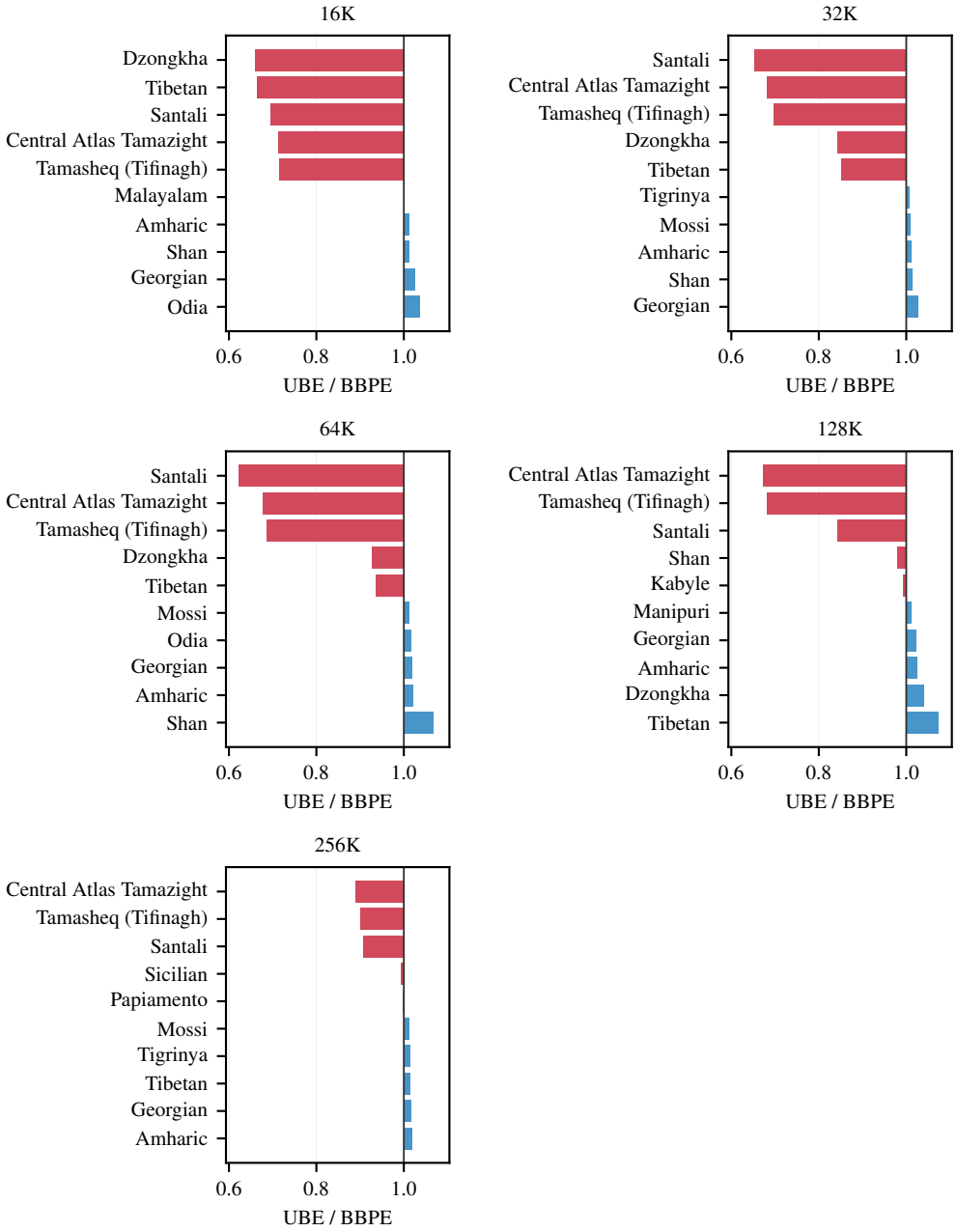

Figure D.3: Best and worst five languages by UBE/BBPE token-count ratio across the full `101-lang` vocabulary sweep (16K–256K). Bars start at the vertical line, which marks equal token counts (ratio 1); red bars mark ratios below 1 (UBE uses fewer tokens) and blue bars ratios of 1 or above.

D.5.2 Per-script-family analysis

Table D.9 breaks down the `101-lang` FLORES-200 results by Unicode script family across all vocabulary sizes (16K–256K). UBE's largest absolute token reductions are in Ol Chiki and Tifinagh at every vocabulary size and in Tibetan up to 64K—all 3-byte UTF-8 scripts with no language-specific mC4 training data. For high-coverage scripts (Latin, Cyrillic, Arabic, Devanagari), UBE changes the family means by less than one token per sentence at every vocabulary size and lowers the Latin, Cyrillic, and Arabic means at every size.

Table D.9: Per-script-family average token counts on FLORES-200 (`101-lang`; 16K–256K; 29 scripts). $\Delta = \text{UBE} - \text{BBPE}$ (negative = fewer UBE tokens).

Script family	Langs	UTF-8	Metric	16K	32K	64K	128K	256K
Latin	127	1–2	UBE	**62.163**	**56.627**	**52.500**	**48.722**	**45.097**
			BBPE	62.373	56.743	52.540	48.751	45.100
			Δ	−0.210	−0.117	−0.040	−0.029	−0.003
Arabic	22	2	UBE	**77.610**	**69.549**	**62.741**	**56.453**	**51.336**
			BBPE	78.312	69.687	62.817	56.489	51.358
			Δ	−0.702	−0.138	−0.076	−0.036	−0.022
Cyrillic	12	2	UBE	**75.391**	**65.828**	**57.626**	**50.389**	**44.300**
			BBPE	75.875	66.044	57.703	50.435	44.315
			Δ	−0.484	−0.216	−0.077	−0.046	−0.015
Hebrew	2	2	UBE	**74.512**	**63.175**	**55.843**	**48.783**	**43.206**
			BBPE	74.965	63.530	55.976	48.797	43.216
			Δ	−0.453	−0.355	−0.133	−0.014	−0.010
Greek	1	2	UBE	**90.479**	**77.161**	**66.125**	**56.125**	**48.226**
			BBPE	90.726	77.342	66.307	56.179	48.243
			Δ	−0.247	−0.181	−0.183	−0.054	−0.017
Armenian	1	2	UBE	**91.947**	**78.250**	**64.792**	**54.347**	**46.934**
			BBPE	92.957	78.649	65.068	54.425	46.963
			Δ	−1.010	−0.399	−0.277	−0.078	−0.030
Devanagari	10	3	UBE	**93.056**	**89.929**	**88.237**	**86.705**	85.871
			BBPE	93.438	90.036	88.247	86.756	**85.869**
			Δ	−0.382	−0.107	−0.010	−0.051	+0.001
Bengali	3	3	UBE	**119.222**	110.351	**104.794**	100.288	96.991
			BBPE	119.609	**110.142**	104.888	**99.876**	**96.733**
			Δ	−0.386	+0.209	−0.095	+0.412	+0.258
Ethiopic	2	3	UBE	**87.370**	75.036	65.440	57.024	49.441
			BBPE	87.440	**74.330**	**64.830**	**56.061**	**48.660**
			Δ	−0.071	+0.706	+0.610	+0.962	+0.781
Tibetan	2	3	UBE	**299.469**	**298.116**	**256.428**	237.486	170.774
			BBPE	452.168	352.053	275.195	**224.709**	**169.761**
			Δ	−152.699	−53.937	−18.767	+12.776	+1.013
Myanmar	2	3	UBE	188.222	173.614	167.616	**139.157**	129.968
			BBPE	**186.831**	**172.278**	**160.686**	141.021	**129.805**
			Δ	+1.391	+1.335	+6.930	−1.865	+0.163
Tifinagh	2	3	UBE	**193.081**	**185.780**	**183.926**	**182.474**	**182.395**
			BBPE	270.173	269.911	269.847	269.815	203.886
			Δ	−77.092	−84.130	−85.921	−87.341	−21.491
Han (Trad.)	2	3	UBE	**49.420**	**43.913**	38.641	34.964	31.997
			BBPE	50.239	44.039	**38.298**	**34.557**	**31.617**
			Δ	−0.819	−0.125	+0.343	+0.408	+0.380
Han (Simp.)	1	3	UBE	**44.910**	39.359	35.015	31.574	28.664
			BBPE	44.920	**39.183**	**34.764**	**31.310**	**28.437**
			Δ	−0.010	+0.176	+0.251	+0.264	+0.227
Hangul	1	3	UBE	**44.092**	**38.633**	**34.112**	30.377	27.495
			BBPE	44.539	38.766	34.193	**30.375**	**27.481**
			Δ	−0.447	−0.132	−0.081	+0.003	+0.014
Japanese	1	3	UBE	**52.419**	46.273	40.783	36.098	32.338
			BBPE	52.668	**46.194**	**40.630**	**35.965**	**32.145**
			Δ	−0.249	+0.079	+0.152	+0.132	+0.193

Table D.9 (continued)

Script family	Langs	UTF-8	Metric	16K	32K	64K	128K	256K
Khmer	1	3	UBE	145.965	**135.804**	**127.642**	**121.825**	118.121
			BBPE	**145.946**	136.047	127.747	121.860	**118.116**
			Δ	+0.020	−0.243	−0.105	−0.035	+0.005
Lao	1	3	UBE	**115.340**	**99.961**	**87.657**	**79.355**	**72.767**
			BBPE	115.791	100.289	88.180	79.768	72.899
			Δ	−0.451	−0.327	−0.523	−0.413	−0.132
Thai	1	3	UBE	**75.640**	69.237	**64.534**	60.990	**58.107**
			BBPE	75.850	**69.173**	64.612	**60.969**	58.122
			Δ	−0.209	+0.064	−0.078	+0.021	−0.015
Telugu	1	3	UBE	**114.800**	**110.878**	105.106	**100.440**	**98.118**
			BBPE	114.948	111.162	**105.057**	100.465	98.162
			Δ	−0.147	−0.284	+0.048	−0.026	−0.044
Tamil	1	3	UBE	**130.214**	**122.921**	**118.113**	**115.209**	**113.794**
			BBPE	130.302	123.013	118.227	115.250	113.797
			Δ	−0.088	−0.092	−0.115	−0.041	−0.003
Kannada	1	3	UBE	121.886	**114.405**	**108.854**	**104.483**	**101.329**
			BBPE	**121.878**	115.094	109.013	104.505	**101.329**
			Δ	+0.008	−0.689	−0.159	−0.022	0.000
Malayalam	1	3	UBE	132.650	126.980	120.760	115.976	**113.153**
			BBPE	**132.444**	**126.831**	**120.714**	**115.951**	113.167
			Δ	+0.207	+0.149	+0.045	+0.026	−0.014
Sinhala	1	3	UBE	**110.344**	**102.037**	**95.809**	91.664	**88.621**
			BBPE	110.516	102.125	95.822	**91.593**	88.624
			Δ	−0.172	−0.088	−0.013	+0.071	−0.003
Odia	1	3	UBE	188.713	**161.316**	135.328	**118.017**	109.633
			BBPE	**182.258**	164.863	**133.265**	**118.017**	**108.809**
			Δ	+6.456	−3.546	+2.063	0.000	+0.824
Gujarati	1	3	UBE	**105.362**	99.345	93.423	**89.804**	86.962
			BBPE	105.835	**98.813**	**93.399**	89.846	**86.943**
			Δ	−0.473	+0.532	+0.024	−0.042	+0.020
Gurmukhi	1	3	UBE	**106.408**	98.947	93.255	**88.974**	**86.336**
			BBPE	107.065	**98.726**	**93.199**	89.055	86.388
			Δ	−0.657	+0.220	+0.056	−0.081	−0.052
Georgian	1	3	UBE	100.074	85.117	71.468	59.405	49.793
			BBPE	**97.688**	**82.745**	**70.135**	**58.003**	**48.937**
			Δ	+2.386	+2.372	+1.333	+1.402	+0.857
Ol Chiki	1	3	UBE	**237.326**	**222.944**	**212.692**	**211.712**	**207.379**
			BBPE	341.312	341.287	341.271	251.125	228.583
			Δ	−103.986	−118.343	−128.579	−39.413	−21.204

D.5.3 Vocabulary utilization diagnostics

Table D.10 reports vocabulary utilization metrics for the four `101-lang/32K` tokenizers. A dead merge is a learned merge token with count exactly zero when the finalized tokenizer retokenizes the frequency-weighted, merged filtered training word table used to produce its training-time `token_counts` metadata. UBE's 512-symbol alphabet does not appear to strand merge capacity: active utilization is 99.8%, all 256 PUA symbols are used, and UBE has fewer dead merge tokens (9) than BBPE (13). Roughly 23.6% of UBE's merges join two pure PUA tokens (UTF-16-path merges), indicating that the UTF-16 path learns productive subword patterns. The higher dead-merge counts for S-UBE and S-BBPE (531/524 vs. 9/13) reflect SuperBPE's permissive cross-word merges, which create long-tail tokens that are not emitted in this final training-time retokenization. Mixed-alphabet results account for 9.8% of merges, including merges with an already mixed operand; the remainder stay entirely within one alphabet. UBE also has roughly twice the share of depth-1 merges, which join two base symbols (17.3% vs. 7.9%), because routed BMP characters can be completed after a single merge from 2 UTF-16 bytes, whereas BBPE typically needs 2 successive merges to collapse a 3-byte UTF-8 decomposition before character-spanning subword merges become available. Depth-1 share is not a standalone quality metric; here, the higher share reflects that the alphabet has already paid much of the floor cost, so fewer hierarchical merges are needed to reach character-level

units. Together with 99.8% active utilization and use of all PUA symbols, this indicates that the extra alphabet capacity is productively spent rather than stranded.

Table D.10: Vocabulary utilization diagnostics (`101-lang/32K`). Other vocabulary sizes are in Table D.11.

Metric	UBE	BBPE	S-UBE	S-BBPE
Alphabet tokens	512	256	512	256
Merge tokens	31,483	31,739	31,483	31,739
Dead merges	9	13	531	524
Active utilization	99.8%	99.9%	98.2%	98.3%
Pure PUA–PUA merges	23.6%	—	24.6%	—
GPT-2–GPT-2 merges	66.6%	—	64.6%	—
Mixed-alphabet merges	9.8%	—	10.8%	—
Depth-1 merges	17.3%	7.9%	19.4%	10.3%
Max merge depth	7	8	14	14

Table D.11: Vocabulary utilization diagnostics across vocabulary sizes (`101-lang`; 16K–256K), extending Table D.10.

Family	Metric	16K	32K	64K	128K	256K
UBE	Dead merges	6	9	16	78	217
	Active util.	99.713%	99.847%	99.911%	99.907%	99.899%
	Pure PUA	24.758%	23.587%	22.297%	20.924%	19.641%
	Mixed alph.	9.082%	9.808%	10.595%	11.846%	13.297%
	Depth-1	24.467%	17.298%	11.715%	7.526%	4.612%
BBPE	Dead merges	9	13	20	82	224
	Active util.	99.825%	99.900%	99.939%	99.921%	99.905%
	Depth-1	12.326%	7.889%	4.937%	2.982%	1.774%
S-UBE	Dead merges	158	531	1,203	3,858	12,251
	Active util.	98.763%	98.216%	98.056%	96.954%	95.198%
	Pure PUA	25.467%	24.607%	23.474%	21.866%	20.902%
	Mixed alph.	9.307%	10.755%	12.696%	13.802%	15.931%
	Depth-1	27.223%	19.423%	13.393%	8.474%	5.225%
S-BBPE	Dead merges	156	524	1,193	3,841	12,203
	Active util.	98.906%	98.303%	98.106%	96.984%	95.225%
	Depth-1	15.655%	10.334%	6.897%	4.233%	2.518%

D.5.4 Cross-dataset robustness by vocabulary size

Table D.12 reports UBE's Gini change across all four evaluation corpora and all available vocabulary sizes under the `101-lang` regime (16K through 256K). For UDHR, SIB-200, and MGSM, $m_\tau(\ell)$ is the mean token count over nonempty UDHR line segments, SIB-200 examples, or MGSM questions, and the same English-normalized non-English ratios enter the Gini. SIB-200 and MGSM contain the same items in every language, so their premiums are content-matched; UDHR entries are not line-aligned, so UDHR premiums are descriptive corpus-level comparisons. UBE's advantage is generally larger on UDHR from 32K onward, while the FLORES-200 reduction is marginally larger at 16K. The NLTK UDHR corpus has 387 entries, nearly twice the number of FLORES-200 languages, and includes many scripts that none of the 101 mC4 training languages uses. Because BBPE tokenizers produce a tokenization for arbitrary Unicode strings rather than emitting unknown tokens [Wang et al., 2020, Radford et al., 2019], we describe such text as byte-fallback-heavy rather than out-of-vocabulary. In these byte-fallback-heavy settings, the encoding-floor reduction is most visible because BBPE has learned few or no compensating merges.

Table D.12: Cross-dataset Gini comparison of UBE and BBPE across vocabulary sizes (`101-lang`; 16K–256K).

Dataset	Vocab	UBE Gini	BBPE Gini	ΔGini
FLORES-200	16K	**0.1927**	0.2194	-12.14%
FLORES-200	32K	**0.1979**	0.2183	-9.36%
FLORES-200	64K	**0.2043**	0.2225	-8.20%
FLORES-200	128K	**0.2140**	0.2258	-5.20%
FLORES-200	256K	**0.2204**	0.2246	-1.86%
UDHR	16K	**0.2158**	0.2449	-11.86%
UDHR	32K	**0.2221**	0.2499	-11.11%
UDHR	64K	**0.2248**	0.2491	-9.74%
UDHR	128K	**0.2322**	0.2552	-9.03%
UDHR	256K	**0.2352**	0.2574	-8.62%
SIB-200	16K	**0.2035**	0.2289	-11.10%
SIB-200	32K	**0.2102**	0.2297	-8.50%
SIB-200	64K	**0.2174**	0.2348	-7.40%
SIB-200	128K	**0.2280**	0.2391	-4.67%
SIB-200	256K	**0.2358**	0.2398	-1.66%
MGSM	16K	0.1710	**0.1699**	+0.65%
MGSM	32K	0.1843	**0.1841**	+0.14%
MGSM	64K	**0.1980**	0.1984	-0.19%
MGSM	128K	**0.2119**	**0.2119**	-0.03%
MGSM	256K	**0.2283**	0.2285	-0.06%

E Statistical and comparative analyses

This section examines statistical reliability, alternative tokenizer designs, Unicode fidelity, and exposure-matched language modeling. Unless stated otherwise, learned-tokenizer rows use a 32K vocabulary and report final BPE output tokens, whereas representation-only rows report base-symbol positions before any learned BPE. We keep those measurement units separate.

E.1 Intrinsic significance analysis

For the N non-English language premiums, Gini is computed as $G = \frac{1}{2N^2\bar{\pi}} \sum_{i,j} |\pi_i - \pi_j|$, where $\bar{\pi}$ is the mean premium. Premium variance uses the sample-corrected $N - 1$ denominator and, unlike Gini, depends on the normalization factor.

For each of the nine prespecified core comparisons, the two-sided Wilcoxon signed-rank test uses 204 paired language-level mean token counts, including English. Holm and Benjamini–Hochberg (BH) corrections are applied jointly across those nine tests. The paired bootstrap resamples the 203 non-English language rows with replacement 10,000 times, applies each sampled index to both tokenizers, and reports the percentile interval for $\mathrm{Gini(UBE)} - \mathrm{Gini(BBPE)}$. Table E.1 shows that all nine Gini intervals lie below zero; the same eight of nine Wilcoxon comparisons remain significant under the raw, Holm-adjusted, and BH-adjusted criteria. The exception is the token-count test at `En-only/64K`; its negative Gini interval is not contradictory because the two procedures estimate different statistics.

E.2 Tokenizer comparisons and composition

The primary-corpus block of Table 5 summarizes the three `101-lang/32K` learned-token comparisons; MYTE uses fixed maps without learned BPE. The SCRIPT-style tokenizers use all 2,273 primary training shards, with no additional per-language cap, and deterministic BPE training under the same script/category boundary policy. Replacing BBPE with UBE lowers their token-premium Gini from 0.2036 to 0.1824; the corresponding standard and SuperBPE-style results are in Tables D.3 and D.4.

Table E.2 partitions all 204 languages into five disjoint script cohorts. South Asian comprises Bengali, Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Odia, Sinhala, Tamil, and Telugu scripts;

Table E.1: Statistical analysis of the nine core FLORES-200 comparisons. The Gini CI uses paired resampling over non-English premiums; Wilcoxon uses paired mean token counts.

Corpus	Vocab	Wilcoxon p	Holm p	BH q	UBE Gini	BBPE Gini	ΔGini 95% CI
En-only	16K	9.21×10^{-6}	2.76×10^{-5}	1.18×10^{-5}	**0.3010**	0.3495	[-0.0616, -0.0323]
En-only	32K	1.85×10^{-3}	3.70×10^{-3}	2.08×10^{-3}	**0.2978**	0.3380	[-0.0537, -0.0244]
En-only	64K	3.19×10^{-1}	3.19×10^{-1}	3.19×10^{-1}	**0.2950**	0.3183	[-0.0353, -0.0109]
6-lang	16K	2.46×10^{-33}	2.21×10^{-32}	2.21×10^{-32}	**0.2703**	0.2960	[-0.0376, -0.0139]
6-lang	32K	4.12×10^{-26}	2.88×10^{-25}	1.24×10^{-25}	**0.2522**	0.2717	[-0.0325, -0.0074]
6-lang	64K	2.34×10^{-22}	1.17×10^{-21}	4.20×10^{-22}	**0.2442**	0.2603	[-0.0299, -0.0044]
101-lang	16K	1.28×10^{-29}	1.02×10^{-28}	5.75×10^{-29}	**0.1927**	0.2194	[-0.0467, -0.0063]
101-lang	32K	1.19×10^{-22}	7.13×10^{-22}	2.68×10^{-22}	**0.1979**	0.2183	[-0.0358, -0.0047]
101-lang	64K	1.54×10^{-20}	6.18×10^{-20}	2.32×10^{-20}	**0.2043**	0.2225	[-0.0364, -0.0012]

Remaining is the complement of the four named cohorts. Each language contains 1,012 sentences, so the overall mean weights cohort means by their language counts. SCRIPT-style's lower overall mean relative to standard boundaries is driven by the South Asian cohort, despite higher English and LC-BMP means (Table 5). Within each boundary policy, the effect of substituting UBE varies across cohorts.

Table E.2: Primary-corpus FLORES-200 breakdown (101-language mC4, 32K vocabularies). Entries are final tokens/sentence; n counts languages.

		Standard		SuperBPE-style		SCRIPT-style	
Script cohort	n	UBE	BBPE	UBE	BBPE	UBE	BBPE
Latin/Arabic/Cyrillic	161	**59.078**	59.205	**64.262**	64.534	**59.719**	59.832
South Asian	21	**103.199**	103.401	91.514	**91.472**	78.385	**77.316**
Han/Japanese	4	43.364	**43.363**	43.465	**43.455**	42.801	**42.722**
Khmer/Lao/Myanmar/Thai	5	130.446	**130.013**	115.093	**113.257**	**124.854**	126.617
Remaining	13	**134.332**	164.495	**130.662**	160.236	**139.682**	170.053
All	204	**69.857**	71.889	**72.137**	74.187	**68.001**	69.957

Tables E.3, E.4, and E.5 separately report capped-corpus tokenizer comparisons, and Table E.6 reports the MYTE base-symbol results.

SCRIPT-BPE. SCRIPT-BPE changes three axes jointly: its block/index representation, script/category pre-tokenization, and character-preserving merge constraint [Land and Arnett, 2025]. UBE changes only the reversible base-symbol representation and uses neither a language identifier nor a Unicode script table. Table E.3 reports the five-tokenizer comparison under the capped-corpus setting. Under SCRIPT-style pre-tokenization, replacing BBPE with UBE lowers token-premium Gini from 0.2048 to 0.1832 (95% paired-language bootstrap CI for UBE minus BBPE: $[-0.038, -0.004]$; 20,000 resamples over 203 non-English languages). The pairwise breakdown in Table E.4 shows that UBE is lower in 169 of the 204 languages in FLORES-200, while SCRIPT-BPE's advantages are concentrated in specific script cohorts. Across five matched tokenizer settings (50 MiB/language at 16K and 32K; 300 MiB/language at 16K, 32K, and 64K), the same 163/204 languages use fewer UBE tokens than SCRIPT-BPE in every setting. On the natural-language input fields of XNLI, XCOPA, and MGSM, UBE is lower for all 25 Latin/Arabic/Cyrillic language–corpus pairs, whereas SCRIPT-BPE is lower for 11/12 other-script pairs.

Boundary-policy comparison. To separate base representation from boundary policy, Table E.5 crosses UBE/BBPE with standard/SuperBPE-style boundaries using the same 101-language mC4 prefix (approximately 50 MiB/language) and 32K vocabulary size as Table E.3. Each SuperBPE-style tokenizer extends the corresponding 16K standard tokenizer to 32K with relaxed boundaries, using the two-stage recipe in Appendix D.3. Both stages use deterministic tie-breaking. Replacing BBPE with UBE lowers every reported quantity within either fixed boundary policy. With SuperBPE-style boundaries, the LC-BMP mean falls by 19.7%, the CJK mean by 3.8%, the overall mean by 2.4%, and Gini by 9.4%.

Table E.3: SCRIPT-BPE comparison on FLORES-200. Means are final BPE tokens/sentence; the CJK and LC-BMP groups are defined in Appendix D.2. Gini uses 203 English-normalized non-English premiums. Tokenizer training uses the same 101-language mC4 prefix (approximately 50 MiB of UTF-8 text per language; 32K vocabulary). Change rows compare UBE with BBPE under the same boundary policy; "Lowest" counts wins among all five methods.

Pre-tokenization	Method	English	CJK	LC-BMP	Overall	Gini	Lowest
Standard GPT-2	UBE (ours)	**36.938**	57.385	**222.054**	66.079	0.2086	**162**/204
Standard GPT-2	BBPE	36.959	58.956	272.617	67.875	0.2281	6/204
Official script/category	SCRIPT-BPE	37.982	**51.069**	223.140	**63.785**	**0.1785**	25/204
SCRIPT-style	UBE (ours)	37.539	56.741	223.497	63.918	0.1832	4/204
SCRIPT-style	BBPE	37.569	58.133	281.154	65.677	0.2048	7/204

UBE change vs. matched BBPE	CJK	LC-BMP	Overall
Standard GPT-2	-2.66%	-18.55%	-2.65%
SCRIPT-style	-2.39%	-20.51%	-2.68%

Table E.4: Pairwise UBE/SCRIPT-BPE win counts (32K tokenizers trained on 101-language mC4 with approximately 50 MiB of UTF-8 text per language).

Evaluation	Script cohort	Comparisons	UBE lower	SCRIPT-BPE lower
FLORES-200	Latin/Arabic/Cyrillic	161	**156**	5
FLORES-200	South Asian scripts	21	1	**20**
FLORES-200	Han/Japanese	4	0	**4**
FLORES-200	Khmer/Lao/Myanmar/Thai	5	0	**5**
FLORES-200	Remaining scripts	13	**12**	1
FLORES-200	All	204	**169**	35
XNLI/XCOPA/MGSM	Latin/Arabic/Cyrillic	25	**25**	0
XNLI/XCOPA/MGSM	Other scripts	12	1	**11**
XNLI/XCOPA/MGSM	All	37	**26**	11

MYTE composition. MYTE is a morphology-driven 256-symbol byte representation learned from language-specific resources and used directly as model input [Limisiewicz et al., 2024]. We apply its released decomposition and morphology-merge maps unchanged, retain the generated code bytes, and route only residual characters through UBE. The compact 512-symbol MYTE-UBE representation shares one 256-symbol region between residual UTF-8 bytes and MYTE code bytes, which are both copied verbatim into the post-MYTE byte stream, and keeps a separate 256-symbol region for routed UTF-16 bytes. No learned BPE is applied to Table E.6; one emitted base symbol is one model-input position. MYTE-UBE reduces the CJK mean by 24.08%, the LC-BMP mean by 29.13%, and total positions by 4.58% relative to MYTE, is lower in 201/204 languages, ties in three, and is higher in none. On all 206,448 FLORES-200 sentences, MYTE-UBE reconstructs the post-MYTE byte stream exactly and matches MYTE's released decoder output. Relative to UTF-8 bytes, UBE alone is shorter in 202/204 languages, ties in two (Catalan and Central Kurdish), and is longer in none. UBE's reductions are largest in the 39 languages in which at least half of the characters of the devtest sentences are 3-byte UTF-8 characters: 29.10% (Korean) to 33.13% (Tibetan), and 31.40% for the mean length of this group. In the other 165 languages, they reach at most 8.30% (Vietnamese). Table I.7 lists all 204 languages; Kim [2026] also reports results for 21 of them, counted on NFC-normalized text. For the 39 languages, MYTE-UBE shortens MYTE's mean by 17.67%. Over the 203 non-English languages, the Gini coefficient of English-normalized base-symbol means is 0.1500 for UBE versus 0.2077 for UTF-8 bytes, and 0.0925 for MYTE-UBE versus 0.1177 for MYTE.

E.3 Unicode fidelity

We audit all 1,112,064 Unicode scalar values and every unique sequence in the Unicode 17 normalization and grapheme-break test files [The Unicode Consortium, 2025] and the emoji test file [Davis and Holbrook, 2025]. For UBE and MYTE, recovery requires $\mathrm{decode}(\mathrm{encode}(x)) = x$; SCRIPT-BPE is also evaluated after its configured NFC normalization and within the character set supported by its released implementation. Across all 1,155,724 suite-level inputs, UBE recovers every raw input

Table E.5: Representation-by-boundary composition on FLORES-200 (101-language mC4, approximately 50 MiB/language, 32K vocabulary). Means are final BPE output tokens per sentence; the CJK and LC-BMP groups are defined in Appendix D.2. Gini uses 203 English-normalized non-English premiums.

Boundary policy	Representation	English	CJK	LC-BMP	Overall	Gini
Standard	UBE (ours)	**36.938**	**57.385**	**222.054**	**66.079**	**0.2086**
Standard	BBPE	36.959	58.956	272.617	67.875	0.2281
SuperBPE-style	UBE (ours)	**39.524**	**58.576**	**189.674**	**67.208**	**0.1736**
SuperBPE-style	BBPE	39.586	60.871	236.155	68.858	0.1916

Table E.6: Representation-only FLORES-200 comparison. Means and totals are base-symbol positions, not learned BPE output tokens; the CJK and LC-BMP groups are defined in Appendix D.2. Gini uses the 203 non-English premiums of these base-symbol means.

Base representation	Alphabet	English	CJK	LC-BMP	Overall	Total positions	Gini
UBE (ours)	512	**130.481**	**91.252**	**277.056**	**176.088**	**36,353,040**	**0.1500**
UTF-8 bytes	256	130.530	133.665	406.632	196.106	40,485,643	0.2077
MYTE-UBE (ours)	512	**109.094**	**94.610**	**247.622**	**141.354**	**29,182,333**	**0.0925**
MYTE	256	109.140	124.612	349.398	148.142	30,583,555	0.1177

and MYTE recovers 1,138,306. SCRIPT-BPE recovers every assigned-through-Unicode-16 scalar in its released valid-character set after NFC, but filters Private-Use and unassigned scalars, and Unicode-17 additions fall outside that released set (Table E.7). Under the stricter raw-identity criterion, SCRIPT-BPE recovers 153,943/155,063 scalars in that valid set; NFC changes the remaining 1,120.

Table E.7: Deterministic Unicode 17 fidelity audit (UBE/SCRIPT-BPE tokenizers from Table E.3). "After NFC" includes SCRIPT-BPE's normalization; no aggregate is reported across excluded inputs. Sequence files are versioned stress tests, not conformance claims.

Unicode 17 input set	Inputs	UBE raw exact	SCRIPT-BPE after NFC	MYTE raw exact
Assigned through Unicode 16 (excluding Private-Use)	155,063	**All recovered**	**All recovered**	154,023/155,063
Assigned in Unicode 17	4,803	**All recovered**	Outside released coverage	**All recovered**
Private-Use (Co)	137,468	**All recovered**	Filtered by design	**All recovered**
Unassigned/reserved (Cn)	814,730	**All recovered**	Filtered by design	**All recovered**
`NormalizationTest.txt` unique inputs	37,671	**All recovered**	166 outside coverage	21,373/37,671
`GraphemeBreakTest.txt` unique inputs	764	**All recovered**	74 outside coverage	684/764
`emoji-test.txt` unique inputs	5,225	**All recovered**	67 outside coverage	**All recovered**
All suite-level cases	1,155,724	**All recovered**	No aggregate across excluded inputs	1,138,306/1,155,724

E.4 Source-byte-matched language-model comparison

Table E.8 compares UBE, BBPE, and BBPE16 at matched cumulative supervised UTF-8 source-byte exposure. All three checkpoints use the same frozen 101-language mC4 corpus and data-loading protocol, 182M architecture, 32K vocabulary, a shared seed, and the same 807,999,969-byte held-out source. The selected BBPE and BBPE16 checkpoints differ from UBE's 361,441,233,445-byte target by less than 0.0005%. UBE's common-text BPB is 0.91% below BBPE's and 0.78% below BBPE16's; BBPE16 requires 18.61% more model-input training tokens to reach the same source-byte exposure. Task-level accuracy leaders vary.

Table E.8: Source-byte-matched language-model comparison. Top: training exposure (steps, model-input positions, supervised UTF-8 bytes) and common-text BPB/BPB Gini. Bottom: zero-shot accuracy (%); English mean weights the five English tasks equally.

Tokenizer	Steps	Model-input positions	Supervised UTF-8 bytes	Common-text BPB↓	BPB Gini↓
UBE (ours, primary tokenizer)	200,000	104,857,600,000	361,441,233,445	**1.5298**	**0.1735**
BBPE	200,437	105,086,713,856	361,439,652,484	1.5439	0.1740
BBPE16	237,228	124,375,793,664	361,440,525,559	1.5418	0.1882

Tokenizer	XNLI	XCOPA	Bele	ARC-E	ARC-C	HSwag	PIQA	LAMB	English mean
UBE (ours, primary tokenizer)	**34.4**	51.8	23.7	**35.9**	17.8	**27.9**	60.1	23.1	**33.0**
BBPE	**34.4**	51.5	**24.0**	35.3	16.8	**27.9**	60.3	**23.3**	32.7
BBPE16	**34.4**	**51.9**	23.3	35.5	**18.0**	27.6	**60.8**	19.5	32.3

E.5 Five-tokenizer language-model comparison at matched exposure

Table E.9 extends the model-level comparison to five 181,688,064-parameter models with 32K vocabularies. Their BPE vocabularies are trained on the same 101-language mC4 prefix (approximately 50 MiB of UTF-8 text per language); all runs share the random seed, 101-language training stream, Transformer blocks, optimizer, evaluation suite, and 807,999,969-byte held-out source. Training steps vary to match UBE's trainer-recorded target of 283,363,416,364 supervised UTF-8 bytes. These tokenizers and the exposure target differ from those in the source-byte-matched comparison above, so BPB values should not be compared across the two protocols.

Multilingual mean equally weights XNLI, XCOPA, and Belebele; English mean equally weights ARC-Easy, ARC-Challenge, HellaSwag, PIQA, and LAMBADA. UBE has the lowest BPB Gini and the highest multilingual and English benchmark-group means, while SCRIPT-BPE has the lowest common-text BPB; UBE's BPB is lower than BBPE and MYTE+BPE. The released 256-symbol MYTE representation is not included directly because matching the target at its observed supervised-byte rate would require approximately 520,943 steps, 2.60× UBE's 200,000-step run. We instead use a derived, parameter-matched 32K BPE variant over the released MYTE representation. Because MYTE+BPE and SCRIPT-BPE can canonicalize or filter inputs, their common-text BPB is representation-conditioned loss per source byte rather than lossless raw-text code length.

Table E.9: Language-model comparison across five tokenizers (182M parameters; 32K vocabularies trained on approximately 50 MiB/language). Single-seed runs match 283.36B supervised UTF-8 bytes. Group means weight benchmarks equally. BPB for MYTE+BPE and SCRIPT-BPE is representation-conditioned; see Appendix E.5 for details.

Method	Common-text BPB↓	BPB Gini↓	Multi. mean↑	English mean↑
UBE (ours)	1.5643	**0.1937**	**36.8**	**33.0**
BBPE	1.5908	0.2006	36.6	32.7
BBPE16	1.5561	0.1996	36.3	32.5
MYTE+BPE (derived 32K)	1.6474	0.2029	36.5	32.2
SCRIPT-BPE	**1.5401**	0.1951	36.4	32.9

E.6 Evaluation coverage and code-heavy inputs

The downstream LM benchmarks XNLI, Belebele, and XCOPA, as well as the tokenizer-level MGSM robustness corpus, include languages whose primary scripts use 3-byte BMP encodings; none of the listed evaluations includes a primary 4-byte non-BMP group (Table E.10). Held-out BPB covers all 101 mC4 languages, including 20 primary 3-byte entries. These counts describe each evaluation's standard language entries rather than the number of LM training languages.

Table E.10: Language-entry coverage by primary encoding-floor group. Each English benchmark denotes ARC-Easy, ARC-Challenge, HellaSwag, PIQA, or LAMBADA separately.

Evaluation	Entries	Primary 3-byte BMP	Primary 4-byte non-BMP
XNLI	15	3	0
Belebele	122	26	0
XCOPA	11	3	0
MGSM	11	5	0
Each English benchmark	1	0	0
Held-out BPB	101	20	0

Because retaining the ASCII floor is central to UBE's design, we also test code-heavy multilingual inputs using 100 SWE-bench tasks [Jimenez et al., 2024] and the ten MAPS translations in the public release [Hofman et al., 2026] (the Chinese translation described in the MAPS paper is not released). Token counts use the primary `101-lang/32K` UBE and BBPE tokenizers and a 32K BBPE16 tokenizer trained on the same corpus. Aggregated within each language over the 100 aligned tasks, UBE uses fewer final BPE tokens than BBPE for English and every one of the ten translations under both input conditions. Table E.11 aggregates the 1,000 non-English inputs; intervals use 10,000 paired bootstrap resamples of the 100 aligned task clusters. UBE still uses 0.18% fewer tokens than BBPE even when complete reference-edit file contents make the inputs 99.24% ASCII bytes, whereas uniform UTF-16 BBPE16 is substantially longer.

Table E.11: Final BPE output-token changes on ten MAPS translations of 100 SWE-bench tasks, using 32K tokenizers trained on the primary 101-language corpus. Negative values favor the named method over BBPE.

Input content	ASCII-byte share	UBE Δ vs. BBPE	BBPE16 Δ vs. BBPE
Issue text	73.91%	**-0.27%** [-0.47%, -0.15%]	+21.92% [+20.56%, +23.19%]
Issue text + edit-file contents	99.24%	**-0.18%** [-0.21%, -0.15%]	+24.86% [+24.35%, +25.40%]

E.7 Corpus-regime sensitivity

Protocol. We compare the six learned-token composition variants in Table 5 across two tokenizer-training regimes: the primary 101-language corpus and an approximately 50 MiB UTF-8-text prefix per language, always at 32K vocabulary size. The cap jointly changes corpus volume and language mixture; it is not an isolated intervention on language balance. Equal byte budgets also do not imply equal semantic content across scripts. The evaluation keeps all 1,012 aligned FLORES-200 devtest sentences in each of the 204 languages fixed. Following the distinction between parallel-content token counts and word-based fertility [Arnett et al., 2025], we report final output tokens per sentence, without added special tokens, and group by the dataset's script tags. These tags apply to whole languages; mixed-script sentences are not split by character. Fixed MYTE maps do not depend on the mC4 cap and remain a separate base-symbol comparison (Table E.6).

Redistribution across scripts. The capped vocabularies use fewer tokens in 181–185 of the 204 languages, depending on boundary policy and representation. The aggregate improvement is therefore not an English or CJK improvement: both groups require more tokens under the cap, whereas the 160 non-English Latin-, Arabic-, and Cyrillic-script languages account for 96–104% of the net decrease in the overall mean. Table E.12 reports every script tag, including the increases for well-covered primary-corpus languages. Within each fixed corpus and boundary policy, substituting UBE for BBPE lowers token-premium Gini by 8.5–10.5%. Its CJK effect is also corpus-dependent: it is near zero under the primary recipe but reduces CJK tokens under every capped boundary policy (Table E.14). The representation benefit thus persists when the primary training recipe is replaced by the capped one.

Why the cap redistributes tokens. The cap changes the training mixture: English falls from 29.2% to 1.0%, Korean, Chinese, and Japanese from 26.5% to 3.0%, and Hindi and Thai from 17.4% to 2.0%, while the remaining 95 languages rise from 26.8% to 94.1%. BPE adds merges in order

of corpus frequency, so a language's share shapes how much of the shared vocabulary serves its text [Conneau and Lample, 2019, NLLB Team, 2024, Hayase et al., 2024]. Consistently, English uses fewer distinct tokens on identical text under the cap (see below), and the fraction of CJK character occurrences split below the character rises from 9.7–13.8% to 34.3–38.7%. Within the LC-BMP group, the Myanmar-script mean token count falls by 6.6–19.3% while the Burmese training share rises from 0.3% to 1.0%. No training language uses Tibetan, Ol Chiki, or Tifinagh as its main script; their script-wise mean token counts change by at most 7.5%. Total volume falls with the mixture, so these shares describe the redistribution rather than isolate its cause.

Table E.12: Mean output tokens per FLORES-200 sentence by script tag for the six composition variants (32K vocabularies), with tokenizers trained on the primary corpus or on the 50 MiB/language prefix. Both rows of a script use the same sentences; n counts languages, and Latin includes English.

			Standard		SuperBPE-style		SCRIPT-style	
FLORES script	n	Corpus	UBE	BBPE	UBE	BBPE	UBE	BBPE
Arabic	22	Primary	**69.549**	69.687	**77.190**	77.862	**70.394**	70.450
		50 MiB	**62.995**	63.039	**68.894**	68.940	**63.335**	63.413
Armenian	1	Primary	**78.250**	78.649	**91.867**	92.875	**79.762**	80.015
		50 MiB	**63.933**	64.063	**73.844**	74.226	**64.669**	64.707
Bengali	3	Primary	110.351	**110.142**	**114.986**	115.807	101.364	**94.897**
		50 MiB	**105.400**	105.439	**104.012**	104.017	88.834	**83.498**
Cyrillic	12	Primary	**65.828**	66.044	**75.168**	75.626	**66.698**	66.912
		50 MiB	**57.978**	58.022	**64.967**	65.036	**58.779**	58.830
Devanagari	10	Primary	**89.929**	90.036	**67.171**	67.198	**58.677**	58.790
		50 MiB	94.952	**94.933**	**85.407**	85.468	74.666	**71.307**
Ethiopic	2	Primary	75.036	**74.330**	**87.280**	87.651	76.581	**75.698**
		50 MiB	65.912	**65.060**	74.994	**73.750**	66.837	**65.781**
Georgian	1	Primary	85.117	**82.745**	100.157	**97.806**	86.253	**83.613**
		50 MiB	73.824	**73.454**	85.892	**84.165**	74.694	**74.425**
Greek	1	Primary	**77.161**	77.342	**90.232**	90.522	**78.051**	78.100
		50 MiB	**67.721**	67.772	**77.343**	77.516	**68.462**	68.509
Gujarati	1	Primary	99.345	**98.813**	**97.409**	97.932	84.574	**81.210**
		50 MiB	93.346	**93.236**	85.798	**83.561**	69.764	**67.950**
Gurmukhi	1	Primary	98.947	**98.726**	**94.967**	95.315	83.107	**82.551**
		50 MiB	**92.489**	92.596	**80.943**	81.054	**68.602**	68.986
Hangul	1	Primary	**38.633**	38.766	**43.622**	44.159	**39.314**	39.381
		50 MiB	**53.995**	55.464	**63.085**	66.668	**54.941**	56.510
Han (simplified)	1	Primary	39.359	**39.183**	37.628	**37.506**	39.681	**39.416**
		50 MiB	**53.877**	54.926	**51.170**	51.963	**53.694**	54.287
Han (traditional)	2	Primary	**43.913**	44.039	**43.901**	43.958	**43.812**	43.836
		50 MiB	**56.774**	59.484	**55.337**	58.238	**56.514**	58.666
Hebrew	2	Primary	**63.175**	63.530	**74.486**	74.938	**64.405**	64.614
		50 MiB	**54.887**	54.912	**61.632**	61.747	**55.925**	56.023
Japanese	1	Primary	46.273	**46.194**	48.428	**48.396**	43.900	**43.797**
		50 MiB	65.507	**65.421**	**67.953**	69.249	**62.043**	62.537
Khmer	1	Primary	**135.804**	136.047	117.627	**116.502**	**112.703**	113.635
		50 MiB	127.890	**127.778**	96.629	**96.395**	**90.826**	93.156
Kannada	1	Primary	**114.405**	115.094	**105.599**	105.614	90.495	**87.544**
		50 MiB	**108.404**	108.449	94.325	**94.241**	74.691	**72.421**
Lao	1	Primary	**99.961**	100.289	**102.399**	103.144	108.217	**103.767**
		50 MiB	**85.275**	85.630	**82.771**	85.199	84.385	**79.887**
Latin	127	Primary	**56.627**	56.743	**60.992**	61.177	**57.210**	57.324
		50 MiB	**52.236**	52.279	**55.379**	55.435	**52.710**	52.749
Malayalam	1	Primary	126.980	**126.831**	96.841	**96.669**	92.365	**88.169**
		50 MiB	123.319	**123.284**	86.055	**85.563**	83.323	**78.650**

Table E.12 (continued)

			Standard		SuperBPE-style		SCRIPT-style	
FLORES script	n	Corpus	UBE	BBPE	UBE	BBPE	UBE	BBPE
Myanmar	2	Primary	173.614	**172.278**	147.969	**143.524**	**173.686**	178.840
		50 MiB	161.930	**160.966**	**119.463**	119.620	**151.898**	159.353
Ol Chiki	1	Primary	**222.944**	341.287	**230.274**	341.440	**242.811**	364.459
		50 MiB	**239.684**	341.934	**242.051**	342.068	**251.140**	364.458
Odia	1	Primary	**161.316**	164.863	187.490	**182.281**	**148.375**	164.074
		50 MiB	172.301	**171.531**	192.737	**187.334**	**152.069**	164.080
Sinhala	1	Primary	**102.037**	102.125	**106.787**	107.439	88.953	**84.163**
		50 MiB	**95.973**	96.134	**95.025**	95.128	74.749	**69.862**
Tamil	1	Primary	**122.921**	123.013	**115.659**	115.820	77.683	**75.619**
		50 MiB	**121.695**	121.851	104.768	**104.531**	73.694	**70.895**
Telugu	1	Primary	**110.878**	111.162	**100.376**	100.439	89.684	**87.714**
		50 MiB	**104.795**	104.852	**85.994**	86.088	73.845	**71.519**
Tifinagh	2	Primary	**185.780**	269.911	**191.536**	270.293	**204.546**	289.680
		50 MiB	**196.078**	270.108	**195.998**	270.280	**204.581**	289.885
Thai	1	Primary	69.237	**69.173**	**59.498**	59.591	**55.979**	58.003
		50 MiB	84.894	**84.610**	79.968	**79.458**	**85.782**	87.061
Tibetan	2	Primary	**298.116**	352.053	**217.926**	275.250	**299.304**	352.571
		50 MiB	**299.339**	352.121	**227.372**	265.607	**300.190**	352.571

Absolute cost and English normalization. Let $m_r(\ell)$ be a language's mean token count under corpus regime r, and $p_r(\ell) = m_r(\ell)/m_r(\text{English})$. Then

$$\log \frac{p_{\text{cap}}(\ell)}{p_{\text{primary}}(\ell)} = \log \frac{m_{\text{cap}}(\ell)}{m_{\text{primary}}(\ell)} - \log \frac{m_{\text{cap}}(\text{English})}{m_{\text{primary}}(\text{English})}. \tag{1}$$

For standard BBPE, the mean non-English premium falls from 2.314 to 1.841, yet 16 non-English languages have lower premiums while using more tokens. Gini is invariant to the common English denominator: its increase from 0.2183 to 0.2281 reflects a change in the distribution of non-English counts, not English normalization. Reporting both absolute counts and dispersion avoids treating a higher English cost as a multilingual compression gain.

Vocabulary use on identical text. The number of distinct token IDs used on the English evaluation sentences falls from 5,864 to 4,504 for standard UBE and from 5,834 to 4,500 for standard BBPE. These are observed active types on the same held-out text, not the number of vocabulary entries exclusively allocated to English. Together with the count changes, they describe a different use of the fixed vocabulary budget; they do not isolate a causal effect of language balance from corpus volume.

Capped-corpus comparisons. Table E.13 adds official SCRIPT-BPE and derived MYTE+BPE to the six composition variants under the same capped training condition. The seven disjoint cohorts cover all 204 languages. South Asian uses the ten script tags listed in Appendix E.2; CJK includes Han (simplified and traditional), Japanese, and Hangul; Southeast Asian includes Khmer, Lao, Myanmar, and Thai. Other is the complement of these named groups. SCRIPT-BPE and MYTE+BPE retain their respective text transforms, whose fidelity is evaluated separately in Table E.7. SCRIPT-BPE's smallest overall mean coexists with higher English and non-English Latin counts than standard UBE, while SCRIPT-style boundaries chiefly reduce South Asian counts. These cohort means expose differences that an overall average alone cannot identify.

Paired uncertainty. For each UBE/BBPE pair, Table E.14 resamples the 1,012 aligned sentence indices jointly across all languages and both tokenizers, with 2,000 bootstrap replicates. Each replicate computes the percentage change in the cohort's mean count. The percentile intervals describe sentence-sampling uncertainty conditional on these saved vocabularies and the fixed language set, not variability across tokenizer retraining. All six English and overall intervals favor UBE. For CJK, all three capped intervals favor UBE, whereas the primary SCRIPT-style pair has a small increase. As in the main LM results, shorter output sequences alone do not establish downstream quality gains [Schmidt et al., 2024].

Table E.13: Capped-corpus output tokens per sentence on disjoint FLORES cohorts (approximately 50 MiB/language; 32K vocabularies). SE Asian denotes Southeast Asian; Other is the residual cohort. Overall means weight cohorts by n.

Method	EN	Other Latin	Arabic/ Cyrillic	South Asian	CJK	SE Asian	Other	All
n	1	126	34	21	5	5	12	204
Standard UBE	36.938	**52.357**	**61.224**	103.716	57.385	124.384	139.800	66.079
Standard BBPE	36.959	52.401	61.269	103.694	58.956	123.990	169.302	67.875
SuperBPE-style UBE	39.524	55.505	67.508	94.845	58.576	**99.659**	**133.260**	67.208
SuperBPE-style BBPE	39.586	55.561	67.562	94.487	60.871	100.058	160.062	68.858
SCRIPT-style UBE	37.539	52.831	61.727	80.186	56.741	112.958	142.836	63.918
SCRIPT-style BBPE	37.569	52.869	61.795	**77.520**	58.133	115.762	175.052	65.677
SCRIPT-BPE	37.982	53.276	62.648	78.162	**51.069**	101.340	144.001	**63.785**
MYTE+BPE	**36.661**	52.893	61.802	104.538	58.360	124.674	159.254	67.764

Why the CJK effect depends on the corpus. UBE's lower floor saves tokens only where BBPE needs three or more byte pieces for a character. Under the primary corpus this happens for 0.2–0.6% of Hangul, Han, and kana occurrences in the CJK sentences, because frequent characters are whole tokens and partial-character byte tokens cover most others in two pieces [Wang et al., 2020]; under the 50 MiB/language cap it happens for 3.5–5.5%, and UBE lowers CJK counts by 2.4–3.8%. UBE also splits more characters whose whole-character token is in its vocabulary (1.7–2.2% of occurrences under the primary corpus, versus 0.1–0.7% for BBPE). UTF-8 distinguishes lead from continuation bytes, but UBE writes both bytes of each UTF-16 code unit with the same 256 PUA symbols, so the high byte of one character and the low byte of the next can form a learned merge, sometimes another character's token (in 典广 they spell 网, U+7F51). Under the primary corpus the CJK mean stays within 0.3% of BBPE; per language, UBE is higher only for Simplified Chinese and Japanese, where BBPE splits the fewest characters into three pieces. SCRIPT's two-symbol encoding likewise matches byte-level BPE on Chinese and Japanese once both are trained with the same pretokenizer [Land and Arnett, 2025].

Table E.14: BBPE-to-UBE token-count changes (%) and 95% paired-sentence bootstrap intervals at fixed corpus and boundary policy. Negative values favor UBE.

Corpus	Boundary	English	CJK	All (204)
Primary	Standard	-0.29 [-0.36, -0.23]	-0.06 [-0.13, +0.01]	-2.83 [-2.84, -2.81]
	SuperBPE-style	-0.51 [-0.60, -0.43]	-0.23 [-0.31, -0.14]	-2.76 [-2.78, -2.75]
	SCRIPT-style	-0.14 [-0.19, -0.10]	+0.12 [+0.05, +0.18]	-2.80 [-2.81, -2.78]
50 MiB	Standard	-0.06 [-0.08, -0.04]	-2.66 [-2.78, -2.55]	-2.65 [-2.66, -2.63]
	SuperBPE-style	-0.16 [-0.20, -0.12]	-3.77 [-3.88, -3.67]	-2.40 [-2.41, -2.38]
	SCRIPT-style	-0.08 [-0.11, -0.05]	-2.39 [-2.49, -2.30]	-2.68 [-2.70, -2.66]

E.8 Emulated commercial tokenizer-training mixtures

Design. The two corpus regimes of Section 3.3 change training volume and language composition together; the corpora below vary composition at a fixed volume. Hayase et al. [2024] infer the language mixtures of commercial tokenizer-training corpora from the order of each tokenizer's first 30K merges. We take their estimates for GPT-4o (o200k_base) from their Table 4 and for Mistral NeMo (Tekken) from their released predictions (percentages rounded to four decimal places). We sum their four English domains into English, drop code because we use no code corpus, and drop languages without an mC4 counterpart (1.6% and 1.1% of the estimated bytes, respectively). After renormalization, the GPT-4o mixture has 43.0% English and 2.9% CJK, and the Mistral NeMo mixture has 43.5% English and 5.7% CJK. The Chinese-centric mixture is a scenario rather than an estimate: following GLM-130B, whose text tokenizer was trained on a bilingual corpus "equally distributed with English and Chinese contents" [Zeng et al., 2023], it gives Chinese 45% and English 45% of the bytes and spreads the remaining 10% in proportion to the GPT-4o mixture's other languages. Table E.15 lists the three mixtures. Each corpus holds 30 GiB of UTF-8 document text taken in source order from mC4 training shards, excluding the held-out shards of Section 3.

A language with too little text repeats its documents to keep its share; this affects only French in the Mistral NeMo mixture (0.87 GiB of repeated text). A second GPT-4o corpus starts after the documents of the first and overlaps it by 3.03 GiB because Spanish, French, Dutch, and Portuguese lack enough disjoint text. We train UBE and BBPE with the pipeline of Appendix B.1 at 32K, at 128K, and at the vocabulary size of a matching public tokenizer: o200k_base (200,000), Tekken (131,072), and Qwen 2.5 (151,643, which the pipeline rounds up to the next multiple of 64, 151,680). All twenty tokenizers are reported below. Evaluation follows Appendix E.7: FLORES-200 devtest, 204 languages, no added special tokens, and 2,000 paired-sentence bootstrap replicates.

Table E.15: Emulated tokenizer-training mixtures. English, CJK, and per-language shares are percentages of UTF-8 bytes; each corpus holds 30 GiB.

Mixture	Basis	Languages	English	CJK	Korean / Chinese / Japanese	Vocabulary sizes
GPT-4o	GPT-4o (o200k_base) [Hayase et al., 2024]	74	43.0	2.9	1.1 / 1.2 / 0.6	32,000, 128,000, 200,000
Mistral NeMo	Mistral NeMo (Tekken) [Hayase et al., 2024]	78	43.5	5.7	3.1 / 0.2 / 2.4	32,000, 128,000, 131,072
Chinese-centric	Scenario after GLM-130B [Zeng et al., 2023]	74	45.0	45.3	0.2 / 45.0 / 0.1	32,000, 128,000, 151,680

Results. Table E.16 reports the BBPE-to-UBE change in each cohort, and Table E.17 gives paired intervals. In every pair, UBE lowers total tokens by 1.9–4.6%, LC-BMP tokens by 12.5–25.3%, English tokens by up to 0.11%, and token-premium Gini by 5.5–11.7%; it also uses fewer tokens in 147–187 of the 204 languages. South Asian changes range from -0.9% to +0.4% and Southeast Asian changes from -4.9% to +2.2%. At 128K and above, UBE still lowers Gini by 5.5–10.0%. CJK effects follow coverage: UBE lowers CJK tokens by 2.3–3.4% at 32K under the English-centric mixtures (0.6% under the Chinese-centric one) but changes them by -0.1% to +1.2% at 128K and above, where BBPE splits at most 0.5% of CJK occurrences into three or more pieces.

Table E.16: BBPE-to-UBE token-count changes (%) on FLORES-200 for the emulated mixtures. Gini is the relative change in token-premium Gini; Lower counts the languages, out of 204, where UBE uses fewer tokens. Negative values favor UBE. The CJK and LC-BMP groups are defined in Appendix D.2.

Mixture	Vocab n	English 1	CJK 5	LC-BMP 7	South Asian 21	SE Asian 5	All (204) 204	Gini	Lower
GPT-4o	32,000	-0.094	-2.341	-17.577	+0.439	+0.416	-3.134	-8.905	180
	128,000	-0.018	+0.160	-18.321	+0.025	+0.804	-2.903	-8.107	176
	200,000	-0.004	+0.538	-12.526	-0.000	-0.185	-1.926	-5.469	170
Mistral NeMo	32,000	-0.068	-3.414	-25.289	+0.165	-4.917	-4.606	-11.681	186
	128,000	-0.015	-0.067	-16.782	+0.012	-0.476	-2.364	-6.010	175
	131,072	-0.022	+0.094	-16.839	-0.168	-0.468	-2.376	-6.044	169
Chinese-centric	32,000	-0.105	-0.625	-18.655	-0.909	-2.510	-3.486	-10.280	187
	128,000	-0.015	+0.816	-20.032	-0.029	+1.265	-3.036	-9.274	175
	151,680	-0.022	+1.175	-20.214	-0.220	-1.471	-3.312	-9.966	185
GPT-4o, 2nd sample	128,000	-0.026	+0.297	-12.933	+0.094	+2.187	-2.105	-5.994	147

Table E.17: BBPE-to-UBE token-count changes (%) with 95% paired-sentence bootstrap intervals for the emulated mixtures. Negative values favor UBE. The CJK and LC-BMP groups are defined in Appendix D.2.

Mixture	Vocab	English	CJK	LC-BMP	All (204)
GPT-4o	32,000	-0.094 [-0.130, -0.062]	-2.341 [-2.443, -2.240]	-17.577 [-17.624, -17.528]	-3.134 [-3.147, -3.121]
	128,000	-0.018 [-0.040, -0.004]	+0.160 [+0.080, +0.232]	-18.321 [-18.383, -18.255]	-2.903 [-2.917, -2.889]
	200,000	-0.004 [-0.011, 0.000]	+0.538 [+0.460, +0.617]	-12.526 [-12.607, -12.440]	-1.926 [-1.938, -1.914]
Mistral NeMo	32,000	-0.068 [-0.100, -0.039]	-3.414 [-3.538, -3.297]	-25.289 [-25.339, -25.236]	-4.606 [-4.625, -4.588]
	128,000	-0.015 [-0.030, -0.004]	-0.067 [-0.144, +0.014]	-16.782 [-16.840, -16.719]	-2.364 [-2.376, -2.351]
	131,072	-0.022 [-0.044, -0.007]	+0.094 [+0.012, +0.177]	-16.839 [-16.898, -16.776]	-2.376 [-2.388, -2.363]
Chinese-centric	32,000	-0.105 [-0.145, -0.069]	-0.625 [-0.734, -0.524]	-18.655 [-18.724, -18.582]	-3.486 [-3.499, -3.471]
	128,000	-0.015 [-0.030, -0.004]	+0.816 [+0.728, +0.898]	-20.032 [-20.109, -19.950]	-3.036 [-3.050, -3.021]
	151,680	-0.022 [-0.041, -0.007]	+1.175 [+1.088, +1.262]	-20.214 [-20.291, -20.133]	-3.312 [-3.326, -3.297]
GPT-4o, 2nd sample	128,000	-0.026 [-0.054, -0.004]	+0.297 [+0.212, +0.375]	-12.933 [-12.997, -12.868]	-2.105 [-2.116, -2.092]

CJK by language. The CJK mean can hide opposite per-language effects (Tables E.18 and E.19). UBE splits no BMP CJK character into more than two pieces, so it saves tokens where BBPE splits characters into three or more. In the Chinese-centric mixture at 32K, BBPE splits 10.7% of Hangul occurrences into three or more pieces, and UBE lowers Korean tokens by 5.4%; Japanese, with 0.07% of occurrences split that far, rises by 3.4%, so the CJK mean falls by only 0.6%. At 128K and above, Korean changes by -0.7% to +0.0%, Traditional Chinese and Cantonese by -0.4% to +0.6%, Simplified Chinese by +0.2% to +0.8%, and Japanese by +0.3% to +3.5%.

Table E.18: BBPE-to-UBE token-count changes (%) with 95% paired-sentence bootstrap intervals for the CJK languages. Negative values favor UBE.

Mixture	Vocab	Korean	Chinese (Simpl.)	Chinese (Trad.)	Cantonese	Japanese
GPT-4o	32,000	-2.19 [-2.37, -2.02]	-1.55 [-1.69, -1.42]	-3.66 [-3.83, -3.49]	-3.43 [-3.60, -3.26]	-1.02 [-1.21, -0.82]
	128,000	-0.71 [-0.84, -0.58]	+0.24 [+0.13, +0.35]	-0.23 [-0.37, -0.10]	-0.16 [-0.31, -0.02]	+1.40 [+1.21, +1.58]
	200,000	-0.48 [-0.62, -0.34]	+0.80 [+0.67, +0.94]	+0.46 [+0.33, +0.58]	+0.45 [+0.33, +0.57]	+1.27 [+1.07, +1.46]
Mistral NeMo	32,000	-0.92 [-1.07, -0.77]	-3.15 [-3.32, -2.97]	-5.31 [-5.51, -5.12]	-5.13 [-5.31, -4.92]	-1.85 [-2.02, -1.69]
	128,000	-0.48 [-0.60, -0.36]	+0.37 [+0.25, +0.50]	-0.37 [-0.51, -0.24]	-0.25 [-0.39, -0.11]	+0.32 [+0.16, +0.48]
	131,072	-0.48 [-0.60, -0.37]	+0.72 [+0.58, +0.86]	-0.25 [-0.39, -0.11]	-0.15 [-0.29, -0.01]	+0.53 [+0.36, +0.70]
Chinese-centric	32,000	-5.36 [-5.59, -5.13]	+0.09 [-0.02, +0.18]	-0.09 [-0.21, +0.03]	-0.15 [-0.26, -0.04]	+3.36 [+3.14, +3.59]
	128,000	-0.23 [-0.42, -0.04]	+0.17 [+0.05, +0.28]	+0.57 [+0.44, +0.71]	+0.50 [+0.37, +0.63]	+2.49 [+2.30, +2.67]
	151,680	+0.04 [-0.14, +0.23]	+0.18 [+0.06, +0.29]	+0.65 [+0.52, +0.78]	+0.50 [+0.38, +0.62]	+3.51 [+3.30, +3.70]
GPT-4o, 2nd sample	128,000	-0.71 [-0.84, -0.59]	+0.37 [+0.26, +0.48]	-0.34 [-0.47, -0.20]	-0.33 [-0.47, -0.20]	+2.15 [+1.95, +2.34]

Table E.19: Share of Hangul, Han, and kana occurrences (%) that BBPE splits into three or more tokens in each CJK language. The UBE share is zero in every pair.

Mixture	Vocab	Korean	Chinese (Simpl.)	Chinese (Trad.)	Cantonese	Japanese
GPT-4o	32,000	2.89	1.88	5.79	5.49	1.57
	128,000	0.44	0.13	0.90	0.88	0.09
	200,000	0.28	0.03	0.22	0.16	0.03
Mistral NeMo	32,000	1.19	5.44	9.56	9.14	2.31
	128,000	0.24	0.25	0.97	0.92	0.06
	131,072	0.24	0.23	0.93	0.87	0.06
Chinese-centric	32,000	10.68	0.01	0.16	0.13	0.07
	128,000	1.58	0.00	0.01	0.01	0.00
	151,680	1.26	0.00	0.00	0.00	0.00
GPT-4o, 2nd sample	128,000	0.47	0.12	0.95	0.94	0.09

Vocabulary composition (exploratory). As an exploratory vocabulary diagnostic, we count entries whose decoded text contains Hangul, Han, or kana (Table E.20). Relative to BBPE, UBE's Hangul entries change by -1.5% to +6.6% and decrease only in the two Chinese-centric pairs at 128K and above, where UBE's Korean change is closest to zero; Han entries change by -2.1% to +1.7%. In every pair, UBE's vocabulary has 10.1–28.4% fewer kana entries than BBPE's. Entries are decoded alone, so cross-code-unit UBE entries need not decode to kana; this census does not measure their use in Japanese text.

Public tokenizers and stability. Table E.21 sets each emulated BBPE beside the public tokenizer of the matching vocabulary size. The emulations reproduce the public tokenizers' CJK and LC-BMP ratios to English within 14%, whereas South Asian ratios differ by up to 68%, and every emulation has fewer single-token Han characters than its public counterpart. Relative to their emulations, o200k_base and Tekken split fewer CJK occurrences into three or more pieces, and Qwen 2.5 splits more (0.52% versus 0.29%). On average, all three public tokenizers use 7.7–10.0 times as many tokens for LC-BMP sentences as for English. In the second GPT-4o sample, the CJK change moves from +0.2% to +0.3% but the LC-BMP change from -18.3% to -12.9%: six of the seven LC-BMP languages are absent from all three mixtures (Burmese is the exception), so their counts depend on incidental merges. The intervals in Tables E.17 and E.18 reflect evaluation-sentence sampling, not variability across tokenizer retraining.

Limitations. The mixtures are estimates from merge order and a reference corpus, not disclosed training data; mC4 stands in for proprietary text, code is excluded, and the analysis measures tokenizers rather than language models.

Table E.20: Vocabulary entries whose text, decoded alone, contains Hangul, Han, or kana characters (exploratory).

		Hangul		Han		Kana	
Mixture	Vocab	UBE	BBPE	UBE	BBPE	UBE	BBPE
GPT-4o	32,000	505	488	587	581	53	74
	128,000	2,251	2,208	2,631	2,643	242	325
	200,000	3,615	3,568	4,190	4,250	435	517
Mistral NeMo	32,000	1,087	1,061	302	297	170	207
	128,000	4,754	4,705	1,718	1,746	870	970
	131,072	4,866	4,822	1,763	1,801	896	997
Chinese-centric	32,000	129	121	8,390	8,310	39	44
	128,000	753	762	40,421	40,513	174	205
	151,680	925	939	49,060	49,147	209	254
GPT-4o, 2nd sample	128,000	2,248	2,210	2,603	2,611	243	338

Table E.21: Emulated BBPE tokenizers beside the public tokenizers of matching vocabulary size. Ratios divide mean FLORES-200 tokens per sentence by the English mean; CJK 3+ is the share of CJK Hangul, Han, and kana occurrences split into three or more tokens; the last column counts distinct characters that a single vocabulary entry decodes to. The CJK and LC-BMP groups are defined in Appendix D.2.

Tokenizer	Vocab	CJK/EN	LC-BMP/EN	South Asian/EN	Chinese (Simpl.)/EN	CJK 3+ (%)	Hangul / Han / kana
GPT-4o BBPE	200,000	1.526	8.383	3.576	1.361	0.14	736 / 2,015 / 141
o200k_base	200,019	1.435	8.892	2.128	1.249	0.08	700 / 2,527 / 153
Mistral NeMo BBPE	131,072	1.627	9.401	3.757	1.668	0.42	818 / 1,141 / 151
Tekken	131,072	1.431	10.026	3.336	1.403	0.20	832 / 1,599 / 156
Chinese-centric BBPE	151,680	1.349	8.455	3.818	0.969	0.29	417 / 5,490 / 132
Qwen 2.5	151,665	1.258	7.692	5.599	1.003	0.52	2,512 / 8,804 / 180

F LM and inference benchmark results

F.1 Full LM benchmark results

Table F.1 reports the complete per-benchmark zero-shot accuracy for the main 182M/32K LM runs and adds the SuperBPE composition rows for UBE and BBPE (S-UBE and S-BBPE) omitted from the main text. Tables F.2, F.3, and F.4 report the corresponding 1.3B, 64K, and 128K variants.

Table F.1: Full LM benchmark results (182M, 32K, 200K mC4 steps). Benchmark columns report zero-shot accuracy. XNLI/Belebele/XCOPA are multilingual; ARC-Easy/ARC-Challenge, HellaSwag, PIQA, and LAMBADA are English-only.

Config	Tok	BPB↓	BPB Gini↓	XNLI	Bele	XCOPA	ARC-E	ARC-C	HSwag	PIQA	LAMB
En-only	UBE	**1.4698**	**0.1989**	**34.4**	23.0	**51.5**	**38.5**	17.0	**27.8**	**62.7**	22.5
En-only	BBPE	1.4909	0.1993	34.3	**23.6**	51.4	38.1	**18.3**	27.5	61.9	**23.6**
6-lang	UBE	1.3760	**0.1880**	**34.9**	23.2	**52.2**	**36.7**	**18.4**	**28.0**	**62.0**	**23.1**
6-lang	BBPE	**1.3680**	0.1930	34.6	**23.5**	52.0	35.6	17.1	27.4	61.3	**23.1**
101-lang	UBE	1.5237	**0.1739**	**34.4**	23.7	51.8	**35.9**	**17.8**	**27.9**	**60.1**	**23.1**
101-lang	BBPE	**1.5174**	0.1762	34.3	**24.8**	**52.0**	35.5	17.0	27.7	59.3	22.8
101-lang (SuperBPE)	S-UBE	**1.4607**	0.1822	34.1	**24.5**	52.1	33.0	17.7	**27.7**	**59.7**	**23.9**
101-lang (SuperBPE)	S-BBPE	1.4876	**0.1802**	**34.3**	24.4	**52.4**	**33.7**	**18.1**	27.6	**59.7**	22.2

Table F.2: LM evaluation (1.3B, 32K vocabulary, 205K Chinchilla-style mC4 steps) across the same three tokenizer-training configurations as Table 4. Lower-coverage rows (`En-only`, `6-lang`) are fixed-compute stress tests. Benchmark columns report zero-shot accuracy.

Config	Tok	BPB↓	BPB Gini↓	XNLI	Bele	XCOPA	ARC-E	ARC-C	HSwag	PIQA	LAMB
En-only	UBE	**1.3668**	**0.1956**	**34.8**	**22.9**	**52.8**	37.3	**19.0**	28.2	62.3	**27.0**
En-only	BBPE	1.4193	0.1971	34.5	**22.9**	52.3	**39.4**	17.4	**28.5**	**62.6**	25.6
6-lang	UBE	1.3106	0.1973	**35.0**	**23.9**	52.6	39.6	18.2	**29.2**	63.0	28.6
6-lang	BBPE	**1.3071**	**0.1956**	34.8	23.1	**52.9**	**40.4**	**18.9**	**29.2**	**63.5**	**28.9**
101-lang	UBE	1.4707	0.1983	34.5	**23.4**	**51.9**	**39.9**	**18.4**	**29.1**	63.2	29.1
101-lang	BBPE	**1.4673**	**0.1972**	**35.0**	23.0	51.1	39.1	18.1	**29.1**	**64.0**	**29.9**

Table F.3: LM evaluation (230M, 64K vocabulary, 200K mC4 steps) across the same three tokenizer-training configurations as Table 4. Benchmark columns report zero-shot accuracy.

Config	Tok	BPB↓	BPB Gini↓	XNLI	Bele	XCOPA	ARC-E	ARC-C	HSwag	PIQA	LAMB
En-only	UBE	**1.4588**	**0.2002**	**34.2**	**23.2**	51.9	**40.2**	**18.4**	**27.6**	**61.9**	**22.6**
En-only	BBPE	1.4879	0.2006	**34.2**	23.0	**52.1**	**40.2**	18.1	**27.6**	61.2	20.6
6-lang	UBE	1.4456	0.1812	34.5	22.9	51.6	**38.1**	**19.3**	27.8	**62.1**	**22.8**
6-lang	BBPE	**1.4329**	**0.1751**	**34.7**	**23.0**	**52.0**	37.7	18.9	**28.0**	61.3	22.4
101-lang	UBE	1.4435	0.2047	**34.8**	**23.4**	**51.9**	38.2	**18.1**	27.6	61.0	**24.7**
101-lang	BBPE	**1.4431**	**0.2037**	**34.8**	23.2	51.5	**39.5**	17.6	**28.1**	**61.4**	23.2

Table F.4: LM evaluation (329M, 128K vocabulary, 200K mC4 steps; `101-lang`). S-UBE and S-BBPE provide SuperBPE-style composition checks. Benchmark columns report zero-shot accuracy.

Config	Tok	BPB↓	BPB Gini↓	XNLI	Bele	XCOPA	ARC-E	ARC-C	HSwag	PIQA	LAMB
101-lang	UBE	**1.5028**	**0.2021**	**34.9**	**23.1**	**51.6**	40.0	**19.4**	**28.7**	62.2	**26.7**
101-lang	BBPE	1.5044	0.2029	34.6	23.0	51.4	**41.0**	**19.4**	28.3	**62.4**	25.5
101-lang	S-UBE	**1.4929**	0.1914	**34.3**	23.4	**52.4**	**39.9**	17.8	28.2	**62.6**	26.7
101-lang	S-BBPE	1.5064	**0.1912**	**34.3**	**24.2**	52.3	39.4	**18.0**	**28.4**	60.8	**29.9**

F.2 Multi-seed stability

To verify that the 182M results are not overly sensitive to random initialization, we repeat the `101-lang/32K` configuration (182M) with 3 independent seeds (Table F.5). Absolute macro BPB varies across seeds, which also change the held-out document order, but the paired same-seed UBE–BBPE gap is small: mean ΔBPB is +0.0046, and the largest absolute same-seed gap is 0.0063. We therefore interpret the multi-seed result as showing no systematic LM-level quality advantage in this setting. At the language level, only Lao differs by more than 0.1 BPB in all 3 runs, with UBE lower by 0.2206–0.4309; Burmese and Sinhala, which regress in the primary run (ΔBPB +0.3504 and +0.2823), differ by at most 0.0096 in runs A and B.

Table F.5: Multi-seed LM evaluation (182M, 32K vocabulary, `101-lang`, 3 seeds). Mean rows report mean ± cross-seed standard deviation for BPB, BPB Gini, and benchmarks. Benchmark columns report zero-shot accuracy. Matching run labels use the same seed.

Language modeling and multilingual accuracy						
Tok	Run	BPB↓	BPB Gini↓	XNLI	Bele	XCOPA
UBE	A	1.4717	0.1879	**34.9**	23.5	**51.8**
	B	1.4747	0.1849	**34.7**	23.4	51.7
	Primary	1.5237	**0.1739**	**34.4**	23.7	51.8
	mean	1.4900±0.0292	0.1822±0.0074	**34.7**±0.3	23.5±0.2	51.8±0.0
BBPE	A	**1.4674**	**0.1863**	34.7	**23.6**	**51.8**
	B	**1.4715**	**0.1833**	34.3	**24.5**	**51.8**
	Primary	**1.5174**	0.1762	34.3	**24.8**	**52.0**
	mean	**1.4854**±0.0278	**0.1819**±0.0052	34.4±0.2	**24.3**±0.7	**51.9**±0.1

English accuracy						
Tok	Run	ARC-E	ARC-C	HSwag	PIQA	LAMB
UBE	A	**37.2**	**17.6**	27.8	**60.9**	24.4
	B	**36.8**	**17.8**	**27.9**	**61.4**	24.0
	Primary	**35.9**	**17.8**	**27.9**	**60.1**	**23.1**
	mean	**36.6**±0.7	**17.7**±0.1	**27.9**±0.1	**60.8**±0.7	23.9±0.7
BBPE	A	36.7	17.4	**27.9**	60.3	**24.7**
	B	36.3	**17.8**	27.7	60.0	**25.1**
	Primary	35.5	17.0	27.7	59.3	22.8
	mean	36.2±0.6	17.4±0.4	27.8±0.1	59.9±0.5	**24.2**±1.2

F.3 Context-window and prompt-processing benchmark

Table F.6 reports same-content prompt-processing measurements for UBE and BBPE. We include this benchmark as a deployment-side check of the main context-window claim: shorter prompts for the same semantic content move byte-fallback-heavy scripts farther from the truncation boundary. We describe the full methodology below, informed by the benchmarking principles of Hoefler and Belli [2015] and Georges et al. [2007].

Hardware and software. All measurements are conducted on a single NVIDIA A100 80 GB GPU with exclusive access, using PyTorch `torch.compile` (Inductor backend, `dynamic=True`), batch size 1, FP32 precision, and single-stream scheduling consistent with the MLPerf Inference single-stream scenario [Reddi et al., 2020]. The `6-lang/32K` tokenizer configuration is used to separate well-covered scripts (en, zh, ko, ja, hi, th) from byte-fallback-heavy scripts (my, am, ar, ru).

Timing and trials. Each language is measured independently with CUDA event timing. Following Hoefler and Belli [2015], we report the timing mechanism and explicitly control for timer overhead, synchronization, warmup, and measurement variability. We execute 5 warmup iterations (to stabilize `torch.compile` code generation, cuDNN autotuning, and GPU thermal state), followed by 30 independent trials of 20 forward passes each for each tokenizer condition. We apply symmetric

Table F.6: Same-content prompt-processing benchmark on 5-sentence FLORES-200 bundles (`6-lang/32K`, 182M, A100 80 GB, batch 1, FP32, `torch.compile`). Token columns are bundle lengths; rates use 30 trials with 95% CI half-widths over 28 samples after symmetric min/max trimming. Δtok = (UBE−BBPE)/BBPE and Δspd = relative sequence-rate change.

Language	UBE tok	BBPE tok	Δtok	UBE rate (seq/s, $\pm$95% CI)	BBPE rate (seq/s, $\pm$95% CI)	Δspd
Amharic	**976**	1,438	-32.1%	**46.0** $\pm$ 0.02	31.8 $\pm$ 0.01	+45.01%
Arabic	**957**	975	-1.8%	**48.1** $\pm$ 0.02	46.1 $\pm$ 0.01	+4.29%
English	**223**	226	-1.3%	**151.0** $\pm$ 0.41	145.3 $\pm$ 0.15	+3.90%
Hindi	**672**	**672**	0.0%	**65.3** $\pm$ 0.02	65.2 $\pm$ 0.03	+0.08%
Japanese	**347**	354	-2.0%	**108.7** $\pm$ 0.05	107.5 $\pm$ 0.05	+1.07%
Korean	**253**	260	-2.7%	**143.9** $\pm$ 0.10	140.1 $\pm$ 0.10	+2.76%
Burmese	**1,074**	1,080	-0.6%	**43.4** $\pm$ 0.01	**43.4** $\pm$ 0.01	+0.01%
Russian	**811**	822	-1.3%	**54.7** $\pm$ 0.02	54.6 $\pm$ 0.01	+0.12%
Thai	**480**	482	-0.4%	**85.9** $\pm$ 0.03	85.5 $\pm$ 0.04	+0.49%
Chinese (Simplified)	278	**276**	+0.7%	**139.5** $\pm$ 0.09	139.4 $\pm$ 0.10	+0.04%

min/max outlier removal (1 from each tail, $\sim$3.3% trim) to guard against cold-cache and scheduling artifacts. Following Georges et al. [2007], we report Student-t 95% confidence intervals computed from the 28 trials that remain after trimming. The primary summary is the trimmed mean $\pm$ 95% CI; the significance tests below use all 30 trials without trimming.

Prompt-processing measurement. We report only prompt processing: each timed trial runs full-model forward passes on an already tokenized prompt held on the GPU, so tokenization, host-to-device transfer, and autoregressive generation are excluded. Transformer prefill contains a self-attention term that scales as $O(n^2 d)$ in sequence length [Vaswani et al., 2017]; I/O-optimized implementations reduce memory traffic but do not remove this sequence-length dependence [Dao et al., 2022]. Prompt processing is therefore the most direct deployment-side measurement we report for tokenizer-induced context-window pressure and truncation risk. Autoregressive decode, by contrast, is often memory-bandwidth-bound [Shazeer, 2019, Pope et al., 2023], with per-step attention cost scaling linearly in key–value cache length, so input-length reductions still help but through a different cost path.

Expected vs. observed speedup. Let $r = n_{\text{UBE}}/n_{\text{BBPE}}$ denote the UBE/BBPE token-count ratio on the same prompt (derivable from the Δtok column in Table F.6). The floating-point operation (FLOP)-only scaling terms $1/r - 1$ for MLP-dominated prefill ($O(nd^2)$) and $1/r^2 - 1$ for attention-dominated prefill ($O(n^2 d)$) provide reference points, but launch overheads, embedding/output layers, and memory effects shift the observed rate. At the width tested here ($d = 768$, 182M scale) and the prompt lengths in Table F.6, the MLP term nd^2 is comparable to or larger than the attention term $n^2 d$ for most languages, so observed rates fall closer to linear-in-n than to quadratic. This matches the data: Amharic (-32.1%, +45.0%) shows the largest speedup; Hindi and Thai, whose token counts barely change, show small but statistically significant speed changes (+0.1%, +0.5%); and English and Arabic deviate from FLOP-only estimates, which we attribute to fixed and kernel-level effects.

Fair comparison metric. We report sequences/second (seq/s) rather than tokens/second. Because different tokenizers produce different token counts for the same text—with disparities up to $15\times$ across languages [Petrov et al., 2023]—tokens/second conflates token count with processing rate. Normalizing by the original text content, following prior content-matched multilingual cost comparisons [Ahia et al., 2023], yields a metric that directly compares how quickly each system ingests the same natural-language input under the same model and hardware budget.

Statistical tests. For each language, we compute Welch's t-test (unpaired, unequal variance) and Mann–Whitney U as a distribution-free check. Welch's test is appropriate because UBE and BBPE are benchmarked as separate timing samples and produce different sequence lengths for the same text. To control for multiple comparisons across 10 languages, we apply Benjamini–Hochberg false discovery rate (FDR) correction at $\alpha = 0.05$. The compact table reports rates and confidence intervals; after FDR correction, 8/10 languages are significant, while Chinese (Simplified) and Burmese are not, matching their near-zero measured speed changes. Each language uses one fixed bundle, so error bars capture run-to-run variability for that input under otherwise fixed hardware and software conditions; residual variance arises primarily from GPU scheduling, thermal drift, and measurement noise.

G Reproducibility and compute disclosure

The UBE tokenizer implementation, the Unicode fidelity audit, and the MYTE-UBE transport and parity-verification code are publicly available at `https://github.com/Samsung/UBE`. The five-way LM comparison in Table E.9 uses private training and evaluation infrastructure; its runner, runtime, and configurations are not part of this release. Raw web-corpus shards are reconstructed from public sources rather than redistributed.

G.1 Tokenizer artifact card

The UBE tokenizer JSONs used in this work are documented as follows:

- **Scope and use.** The paper's FLORES-200 evaluations and LM runs; each artifact is a standard Hugging Face `tokenizer.json` with unchanged BPE merge semantics.
- **Data and procedure.** Fixed-capped mC4 subsets and two-phase BPE training are documented in Appendix Sections B.1 and B.3.
- **Evaluation and limitations.** Cross-dataset robustness, LM evaluation, and per-language FLORES-200 tables are reported in Appendix Sections D.5.4, F.1, and I.2; limitations are collected in Appendix H.4.
- **License.** The UBE implementation builds on the Hugging Face `tokenizers` library, whose code keeps its Apache 2.0 license; our modifications and additions are licensed under CC BY-NC-SA 4.0 (Table I.2).

G.2 Compute disclosure

The primary 28-run LM suite and its single-GPU evaluations ran on internal Linux nodes with NVIDIA A100 80 GB GPUs. Tokenizer training used our two-phase CPU pipeline. Table G.1 gives approximate active compute for this suite only; it excludes the additional comparisons in Tables E.8 and E.9. For LM rows, GPU-hours per run are wall-clock hours multiplied by 8 GPUs; wall-clock ranges are coarse estimates adjusted for model size and exclude queueing, download, preprocessing, and checkpoint-transfer time.

Table G.1: Approximate active compute budget for the primary 28-run LM suite. GPU-hours per run are wall-clock hours multiplied by 8 GPUs for LM training rows.

Stage	Configured steps	Runs	Wall-clock/run	GPU-hours/run	Total GPU-hours
182M/32K LM pre-training	200,000	12	1–2 days	192–384	2,304–4,608
230M/64K LM pre-training	200,000	6	1.5–3 days	288–576	1,728–3,456
329M/128K LM pre-training	200,000	4	2–4 days	384–768	1,536–3,072
1.3B/32K LM pre-training	205,000	6	8–14 days	1,536–2,688	9,216–16,128
Downstream eval. + inference	n/a	agg.	<6 h	<6	<6

Multiplying the primary-suite per-run ranges by their run counts gives approximately 14.8K–27.3K A100 80 GB GPU-hours, plus <6 GPU-hours for downstream evaluation and inference benchmarks. The CPU-only tokenizer matrix used about 14 hours wall-clock and 180 CPU-worker-hours; intrinsic evaluation, contamination checks, and statistics used about 3 additional CPU-hours. The primary-corpus SCRIPT-style vocabulary construction additionally took 4.09 hours for UBE and 9.23 hours for BBPE using eight CPU threads, excluding the preceding collection of corpus statistics.

Table E.8 reuses the primary UBE checkpoint and adds two training runs: BBPE on H100 and BBPE16 on A100. Table G.2 reports the recorded training times for the five additional runs underlying Table E.9. The available records do not establish complete accelerator-time accounting for these additional comparisons, so they are not folded into the primary-suite A100 estimate.

Table G.2: Recorded training wall-clock time for the five-tokenizer matched-exposure comparison (Table E.9), separate from the primary-suite budget.

Method	Steps	Training time (hours)
UBE	200,000	12.94
BBPE	178,549	24.76
BBPE16	177,699	24.68
MYTE+BPE	179,526	12.06
SCRIPT-BPE	170,191	11.39

H Positioning, related work, and limitations

Table H.1 summarizes the experimental scope of selected tokenizer studies. UBE combines tokenizer training on 101 languages, intrinsic evaluation across 204 languages, and a primary suite of 28 LM-training runs. For Abagyan et al. [2026], the two multilingual benchmarks are Belebele and M-MMLU; Aya denotes open-ended generation, and MT denotes FLORES-200 translation (BLEU).

Table H.1: Experimental scope of selected tokenizer studies. Tok. langs = tokenizer-training languages, eval langs = intrinsic-evaluation languages, and benchmarks = downstream benchmarks. NR = intrinsic-evaluation language count not separately reported; mono/multi = monolingual/multilingual. The UBE row covers the primary 28-run suite.

Paper	Venue	Tok. langs	Eval langs	LM size	LM tokens	Benchmarks	Vocab
Arnett+	NeurIPS'25	97	97	—	—	—	8–262K
Schmidt+	EMNLP'24	1 (en)	1	350M–2.4B (64 runs)	200B	10 EN	33–49K
SuperBPE	COLM'25	1 (en)	1	680M–11B	∼330B (8B baseline)	30 EN	200K
Abagyan+	ACL'26	62	NR	3.3B	100B+10.5B	2 multi+11 EN + Aya + MT	100–250K
Land & Arnett	TokShop'25	12 (mono)	12 (mono)	—	—	—	64K (mono); 256K merges (multi)
MAGNET	NeurIPS'24	9	9	126M	∼10B bytes	4 multi+2	50–250K (BPE baselines)
UBE (ours)	this work	101	204	182M–1.35B (28 runs)	105B+27B	3 multi+5 EN	16–256K

Sources: Arnett+ [Arnett et al., 2025]; Schmidt+ [Schmidt et al., 2024]; SuperBPE [Liu et al., 2025]; Abagyan+ [Abagyan et al., 2026]; Land & Arnett [Land and Arnett, 2025]; MAGNET [Ahia et al., 2024]. X+ denotes X et al.

H.1 Public large language model tokenizer comparison

Table H.2 reports the FLORES-200 Gini coefficient and total token count for six tokenizers used by public LLMs, corresponding to the reference points in Figure 2a. These tokenizers span a wide vocabulary range (100K–248K). The Qwen 2.5 row uses the tokenizer released with Qwen2.5-7B, which is identical to the Qwen2.5-3B tokenizer; the Qwen 3.6 row follows the official model-card/configuration metadata for the larger Qwen3.6-27B [Qwen Team, 2026] multimodal tokenizer and its special-token metadata. All measurements use the same 204-language FLORES-200 devtest evaluation as the paper's intrinsic benchmarks.

H.2 Representative compact language models and tokenizer scales

Public papers, technical reports, and model cards rarely disclose the exact number of languages used for tokenizer training. Table H.3 therefore reports the public language scope stated in official papers, technical reports, model cards, blogs, or configs rather than the tokenizer-training language count, and should be read as a deployment-facing survey rather than a like-for-like tokenizer-training comparison. When a row spans a model family, the Scale column reports the public training-token range across the listed parameter sizes. The pattern is nevertheless informative. Many English-centric compact models

Table H.2: FLORES-200 Gini coefficient and total token count for tokenizers used by public LLMs, included as Figure 2a reference points.

Tokenizer	Vocab	EN avg	Total (M)	Gini	Type
GPT-4o (o200k)	200,019	26.554	11.6	0.2323	tiktoken
GPT-4 (cl100k)	100,277	26.860	18.6	0.3287	tiktoken
Llama 3	128,256	26.849	16.6	0.3318	tiktoken-BPE
Qwen 2.5	151,665	27.293	15.4	0.2724	BPE
Qwen 3.6 multimodal tokenizer[†]	248,044	27.204	12.8	0.2420	BPE
Mistral NeMo	131,072	27.497	14.2	0.3030	Tekken

[†] The Qwen 3.6 multimodal tokenizer is taken from the `Qwen/Qwen3.6-27B` release. The official configuration reports a padded `vocab_size` of 248,320, while this table reports the vocabulary size measured from the public tokenizer files (248,044); the row is therefore an upper-end reference point and is not directly comparable to the Qwen 2.5 text-model row.
Tokenizer citations: tiktoken [OpenAI, 2023]; Llama 3 [Grattafiori et al., 2024]; Qwen 2.5 report [Qwen Team, 2025] and release blog [Qwen Team, 2024]; Qwen 3.6 release blog [Qwen Team, 2026]; Mistral NeMo release blog [Mistral AI, 2024a] and model card [Mistral AI, 2024b]. Vocabulary sizes are measured from the public tokenizer files.

still operate with 32K–50K vocabularies, while compact public multilingual models either remain 32K-class (Phi-3.5-mini) or jump to much larger family tokenizers (Llama 3.2, Qwen 2.5). This is the regime split most relevant to UBE: small or moderate vocabularies leave less room to absorb script-specific merge coverage, whereas very large multilingual vocabularies partially compensate through scale alone. Table H.3 compiles representative public examples; sources are listed below the table to identify the source for each row's language scope, vocabulary scale, and training scale.

Table H.3: Representative compact public LMs grouped by tokenizer vocabulary scale: compact (32K–50K), large (100K+), and family-inherited (128K+). Tok. source identifies the tokenizer source or family.

Model	Lang scope	Params	Vocab	Tok. source	Scale
Compact tokenizer (32K–50K)					
SmolLM2	English	135M–1.7B	49,152	self-trained	2T–11T tok
MobileLLM	English	125M–1B	32,000	Llama 2	1T tok
Pythia	English	160M–2.8B	50,277	GPT-NeoX	300B tok
TinyLlama	English + code	1.1B	32,000	Llama 2	3T tok
Phi-3.5-mini	23 langs	3.8B	32,064	Llama 2	3.4T tok
Large tokenizer (100K+)					
StableLM 2	7 langs	1.6B	100,352	Arcade100k (cl100k ext.)	2T tok
Phi-4-mini	23 langs	3.8B	200,064	o200k	5T tok
Inherited from larger model (128K+)					
Llama 3.2	8 langs	1B–3B	128,256	Llama 3.1	9T tok
SmolLM3	6 langs	3B	128,256	Llama 3.2	11T tok
Qwen 2.5	29+ langs	0.5B–3B	151,936*	Qwen family	18T tok
BLOOM	46 langs	560M–3B	250,680	BLOOM-176B	341B tok

* Padded `vocab_size` from the `Qwen/Qwen2.5-3B` `config.json` (`https://huggingface.co/Qwen/Qwen2.5-3B`); the Qwen 2.5 technical report [Qwen Team, 2025] reports 151,643 regular tokens plus 22 control tokens (151,665 in total). Pythia's configuration pads its 50,277-token vocabulary to 50,304.
Sources: SmolLM2 [Allal et al., 2025]; MobileLLM [Liu et al., 2024]; Pythia [Biderman et al., 2023]; TinyLlama [Zhang et al., 2024]; Phi-3.5-mini [Microsoft, 2024]; StableLM 2 [Bellagente et al., 2024]; Phi-4-mini [Abouelenin et al., 2025] and its model card [Microsoft, 2025]; Llama 3.2 [Meta, 2024] and the Llama 3 report [Grattafiori et al., 2024]; SmolLM3 [Bakouch et al., 2025] and its model card [HuggingFaceTB, 2025]; Qwen 2.5 [Qwen Team, 2025] and its release blog [Qwen Team, 2024]; BLOOM [BigScience Workshop et al., 2023].

H.3 Extended related work

Measuring tokenization disparity. Petrov et al. [2023] show on parallel text that tokenized length can vary sharply across languages, and Ahia et al. [2023] show that the same content-matched disparity appears in API pricing; we therefore use aligned sentence-level token premiums as our main intrinsic measure of cross-lingual token-budget disparity (Section 3). Follow-up work links tokenization disparity to unequal utility, downstream performance, latency, and infrastructure/access costs [Lundin et al., 2026, Dixit and Dixit, 2026, Teklehaymanot and Nejdl, 2025]. Arnett et al. [2025] analyze causes and mitigations of cross-lingual tokenizer inequity, while Shani et al. [2026] situate tokenizer choices among broader design choices behind multilingual performance gaps. Related

metrics quantify disparity at three levels: tokenizer (Single Token Retention Rate [Nayeem et al., 2025]), downstream LM task (Performance Realization Ratio [Hu et al., 2025]), and dialectal NLP task (Tokenization Parity [Petrov et al., 2023], as applied by Kanjirangat et al. [2025]). Alqahtani et al. [2026] argue for tokenizer–model co-design and standardized evaluation. Arnett et al. [2024] define the byte premium to compare dataset sizes across languages; Appendix B.3 relates this corpus-level measure to the encoding floor. Larger vocabularies absorb more frequent patterns, although added tokens become progressively less frequent [Reddy et al., 2025], and token-premium disparities persist across comparably trained monolingual tokenizers even as vocabulary size increases [Arnett et al., 2025]; spans that no learned merge covers still pay the full encoding floor, motivating our aligned token premiums and English-normalized dispersion metrics.

Raw-text tokenizers. SentencePiece supports BPE and unigram segmentation directly from raw text [Kudo and Richardson, 2018]; it is not restricted to byte-based representations.

Script-specific tokenization. A growing cluster of work addresses tokenization for specific script families. Rana et al. [2026] propose MUTANT, a recipe for multilingual tokenizer design with language-aware pre-tokenization and subword/multiword-aware training; their MUTANT-Indic tokenizer improves average fertility by 39.5% over Llama 4 across English, 22 Indian languages, and code. Brahma et al. [2025] propose Constrained BPE, which enforces script-specific constraints for Indic scripts—keeping dependent vowels cohesive with base consonants. Velayuthan and Sarveswaran [2025] replace byte-level character extraction with grapheme-aware units, benefiting abugida scripts whose written units (graphemes) span multiple code points. Thakur et al. [2025] systematically study vocabulary size, pre-tokenization rules, and training-corpus composition for Indic scripts, linking these choices to both token-to-word efficiency and model quality. More recently, Darshana [2026] proposes a three-layer syllable-aware architecture that separates script-specific segmentation from BPE compression and reports large token reductions for Sinhala and Hindi. Such solutions are increasingly common, but each is tailored to specific scripts or languages; UBE instead applies one reversible, language-agnostic routing rule based on UTF-8 byte length.

Byte-level model architectures. An alternative strategy is to reduce reliance on a fixed tokenizer altogether. CANINE [Clark et al., 2022] operates directly on Unicode characters, and ByT5 [Xue et al., 2022] operates on UTF-8 bytes. MAGNET [Ahia et al., 2024] learns adaptive segmentation boundaries within the model. SpaceByte [Slagle, 2024] and BLT [Pagnoni et al., 2025] show that raw-byte language modeling can become competitive when the architecture allocates computation unevenly across a byte stream. More recently, Bolmo [Minixhofer et al., 2026] introduces a family of open byte-level LMs using a two-stage "byteification" procedure that converts subword models into byte-level models with minimal additional training. FLEXITOKENS [Owodunni et al., 2026] uses learnable boundary predictors between bytes, reducing over-fragmentation and improving performance on token-classification and generative tasks. H-Net [Hwang et al., 2026] learns content- and context-dependent byte segmentation jointly with the model, outperforming BPE-tokenized Transformers with especially large gains for Chinese and code. MrT5 [Kallini et al., 2025] adds a learned delete gate to ByT5 that dynamically shortens byte sequences, adapting compression rates per language. These approaches move segmentation into the model, but they change its architecture or input granularity and are generally not direct replacements in subword-tokenizer pipelines. Token reduction removes or consolidates tokens, for example by pruning, merging, or distillation, to shorten the token sequence that vision, language, and multimodal generative models process, and Kong et al. [2026] argue that it should go beyond efficiency; it is complementary to UBE, which changes only the base symbols that BPE starts from. Replacing UTF-8 with UBE gives byte-level models two input positions per 3-byte BMP character rather than three (Table E.6). Under UBE's low-byte-first order, the leading symbol of a 3-byte BMP letter or dependent mark identifies the character among the 256 code points that share its high byte, so a predictor that knows this high byte can tell letters from marks with one-symbol lookahead [Kim, 2026]. In that work, small causal predictors given one symbol of lookahead reach 0.06% balanced grapheme-boundary error on UBE input versus 8.76% on UTF-8 input, evaluated at code point boundaries in 17 Brahmic-script languages of FLORES-200.

H.4 Additional scope and limitations

The following limitations supplement the core scope statement in the main text.

Potential societal impacts. The intended positive impact of UBE is more equitable multilingual access: by lowering token budgets for scripts that are structurally penalized under UTF-8 BBPE, the method can increase usable context and reduce truncation pressure, compute cost, and latency disparities for users whose languages rely on 3-byte BMP UTF-8 encodings and remain close to the byte fallback floor. The main negative risk is that the same context-capacity gains can be folded into more capable multilingual LMs, including systems that may be deployed for surveillance, misinformation, or other harmful applications. We therefore frame UBE as a tokenizer-level context-capacity and allocative-fairness intervention rather than as a complete solution to broader social fairness or misuse risks.

Evaluation methodology. Our primary evaluation relies on translated parallel text (FLORES-200), which may exhibit translationese artifacts; we check consistency on three additional corpora (UDHR, SIB-200, MGSM). All premiums use per-language mean token counts; SIB-200 and MGSM are content-matched, whereas UDHR lines are not aligned across languages. Token premiums are computed relative to English throughout, and both Gini and variance exclude the English reference row. For a fixed set of non-English languages, changing only the common normalization factor leaves Gini unchanged; it does not imply invariance to pairwise-ratio aggregation.

Low-coverage training regimes. UBE's dual alphabet reserves 256 extra initial symbols relative to BBPE, so at very narrow tokenizer-training coverage and small vocabularies (the `En-only/16K` stress regime) UBE wins in only 37 of 204 languages; the alphabet-separation trade-off discussion in Section 3.1 notes that the absolute per-language regressions are small (median 0.090 tokens per sentence) and that Gini still falls at `En-only/16K` (-13.9%), but users of very-narrow-coverage compact tokenizers should expect a reduced per-language win rate rather than a uniform improvement.

Downstream coverage for most-affected scripts. The largest intrinsic token-count reductions are on low-coverage 3-byte scripts such as Santali, Central Atlas Tamazight, and Tibetan. These languages are not included in the 101-language mC4 LM corpus used for downstream training and evaluation, so the downstream experiments test the tokenizer mechanism indirectly rather than measuring gains for those exact languages.

SuperBPE composition. Because the SuperBPE composition changes both encoding substrate and pre-tokenization, we interpret the interaction analysis narrowly. In the 2×2 factorial design, the interaction term is bounded but sign-changing (-1.1% on average across 16K–256K; per-vocabulary values in Appendix D.3), supporting qualitative complementarity rather than uniform additivity in the setting we test. Our SuperBPE re-implementation follows the reference recipe [Liu et al., 2025]: the permissive Stage-2 regex of the authors' released training scripts, the four-word cap on merged tokens (Appendix A.1.2 of that paper), and the colon-space block (Appendix A.1.4).

Cross-script byte sharing. UBE's dual-alphabet design forgoes a form of coincidental sharing available to BBPE's single UTF-8 byte alphabet: BBPE can sometimes learn merges that reuse shared byte values across scripts or around universal characters such as whitespace and punctuation. This occasionally helps high-coverage languages like Georgian and Amharic, whereas UBE keeps UTF-8-path and UTF-16-path bytes distinct to preserve marker-free decoding (see the alphabet-separation trade-off discussion in Section 3.1); future work could explore selective sharing for a small set of universal characters to mitigate these regressions while preserving the disjoint-alphabet decoding guarantee.

Pre-tokenizer dependence. The core intrinsic matrix uses the GPT-2 regex pre-tokenizer; results under other boundary policies may differ. The SuperBPE- and SCRIPT-style matched comparisons in Appendix E show that replacing BBPE with UBE still reduces overall, English, and LC-BMP token counts under two substantially different pre-tokenization policies, but this does not establish invariance to all possible boundary rules.

Character routing and grapheme clusters. UBE routes characters independently; it is not a grapheme-aware tokenizer and does not normalize canonically equivalent strings beyond the configured upstream pre-tokenizer/normalizer. Scripts whose written units span multiple characters, zero-width joiner sequences, and combining-mark sequences may therefore still require multiple routed units even when each character follows the intended encoding path.

Comparison scope. The downstream LM experiments focus on isolating the byte-substrate change within otherwise matched training and evaluation pipelines. Appendix E includes tokenizer-level SCRIPT-BPE comparisons, a representation-only MYTE composition, a source-byte-matched LM comparison including BBPE16, and a separate matched-exposure model-level comparison with SCRIPT-BPE and a derived 32K MYTE+BPE variant. The latter changes several tokenizer design axes jointly and uses one matched seed, so we interpret it as a comparison across tokenizer designs rather than causal isolation of the representation. Direct model-level comparisons with the released 256-symbol MYTE itself [Limisiewicz et al., 2024] and Bit-level BPE [Moon et al., 2025] remain future work.

Emulated commercial mixtures. The analysis in Appendix E.8 emulates inferred mixtures on mC4 rather than proprietary data and evaluates tokenizers only; its intervals cover evaluation sentences, and a second GPT-4o 128K corpus sample shifts the LC-BMP token-count reduction by 5.4 percentage points.

Non-BMP and symbol-heavy text. Because 4-byte UTF-8 code points remain 4 bytes under UTF-16 surrogate pairs, UBE provides no encoding-floor reduction for supplementary-plane characters. We have not evaluated UBE on symbol-heavy social-media or chat domains, so we do not claim robustness for those settings.

Streaming and constrained decoding. The dual-buffer decoder's streaming and constrained-decoding requirements are detailed in Appendix A.4; our implementation supports both modes, but we have not benchmarked constrained-decoding throughput.

Matched shard-local singleton filtering. The two-phase distributed BPE pipeline applies the same shard-local singleton filtering before global merge selection for both UBE and BBPE. Because filtering operates at the word level—and both tokenizers share the same Stage 1 regex split, word boundaries, and per-word frequencies on the same text—it removes the same raw word types in both conditions; only the byte-level representation of retained words differs. The filtering step therefore does not act as a confound between UBE and BBPE.

I Appendix long tables

I.1 Existing assets and licenses

Tables I.1 and I.2 summarize the external datasets and software assets used in the paper, together with the license or usage constraint that governed their use in our experiments. To keep long tables at the very end of the appendix, we collect those inventories here. We do not redistribute web-scale corpora such as mC4; our experiments use subsets reconstructed locally from the public sources.

Table I.1: Existing dataset and benchmark assets used in this work.

Asset	Access point / version used	License	Terms / handling in this work
mC4 pretraining corpus	Hugging Face `allenai/c4` multilingual; deterministic 101-language local Arrow mirror	ODC-BY	Derived from Common Crawl snapshots. Fixed capped subsets are used for tokenizer and LM training; the local Arrow mirror is reconstructed from public sources and not redistributed.
FLORES-200	Hugging Face `facebook/flores`, devtest split	CC BY-SA 4.0	Used for the primary parallel intrinsic benchmark.
UDHR	NLTK `udhr2` corpus (Unicode UDHR) and United Nations declaration text	Public UN text; public corpus distribution	Used only for robustness evaluation. Our loader downloads the public corpus when absent and records the local corpus checksum.

Continued on next page

Table I.1 – continued from previous page

Asset	Access point / version used	License	Terms / handling in this work
SIB-200	Hugging Face `Davlan/sib200`, test split	CC BY-SA 4.0	Used only for robustness evaluation. Our loader uses a pinned public revision and records a text checksum for the measured split.
MGSM	Hugging Face `juletxara/mgsm`, test split	CC BY-SA 4.0	Used for robustness evaluation. Our preprocessing script builds the evaluation JSON with the original question text from the public dataset.
XNLI	`facebookresearch/XNLI`; Hugging Face `facebook/xnli`, test split	CC BY-NC 4.0	Used for multilingual downstream evaluation and the input token counts of Table E.4 only. We preserve the noncommercial restriction and do not redistribute the data.
XCOPA	Hugging Face `cambridgeltl/xcopa`, test split	CC BY 4.0	Used for multilingual downstream evaluation and the input token counts of Table E.4 only. Our preprocessing script regenerates the local evaluation JSON from the public dataset.
Belebele	Hugging Face `facebook/belebele`, test split	CC BY-SA 4.0	Used for multilingual downstream evaluation only.
ARC-Easy/ARC-Challenge	Hugging Face `allenai/ai2_arc`, test split for reported metrics	CC BY-SA 4.0	Used for English downstream evaluation only.
HellaSwag	Hugging Face `Rowan/hellaswag`, validation split for reported metrics	MIT	Used for English downstream evaluation only.
PIQA	Hugging Face `ybisk/piqa`, validation split for reported metrics	AFL-3.0	Used for English downstream evaluation only.
LAMBADA	Hugging Face `EleutherAI/lambada_openai`, test split	Modified MIT	OpenAI-preprocessed LAMBADA; the original release is CC BY 4.0. Used for English downstream evaluation only.
MAPS (SWE-bench subset)	Hugging Face `Fujitsu-FRE/MAPS`, rev. `2b29573`, with task files at each base commit	CC BY 4.0; repository licenses	Token counts for Table E.11 only; not redistributed.
Unicode 17.0 data files	`unicode.org/Public/17.0.0`	Unicode License v3	Fidelity audit (Table E.7) only.
Ethnologue speaker counts	Ethnologue 27th edition (citation-only lookup)	Proprietary reference work	Used only for the speaker-population annotations in Table C.1. No Ethnologue content is redistributed.

Table I.2: Existing software assets used in this work.

Asset	Access point / version used	License	Terms / handling in this work
Hugging Face `tokenizers`	Upstream `huggingface/tokenizers`; 0.22.3-dev.778/779	Apache 2.0	Primary SCRIPT-style tokenizer training uses dev.779. Our modifications and additions are licensed under CC BY-NC-SA 4.0; the upstream code keeps its Apache 2.0 license and attribution files.
LitGPT	`Lightning-AI/litgpt`, version 0.5.11	Apache 2.0	Our trainer vendors and modifies selected LitGPT modules while preserving the upstream license notice and attribution.
lm-evaluation-harness	EleutherAI repo, version 0.4.13	MIT	Used as the evaluation backend for our offline benchmark runner, pinned to this version.
SCRIPT-BPE	`sanderland/script_tok`, commit `54a1058`	Apache 2.0	Official SCRIPT-BPE tokenizers in Tables E.3, E.4, E.7, E.9, and E.13.

Continued on next page

Table I.2 – continued from previous page

Asset	Access point / version used	License	Terms / handling in this work
MYTE maps and decoder	`tomlimi/MYTE`, commit `177299c`	Not stated in the repository	Used unmodified for Tables 5, E.6, E.7, E.9, and E.13; not redistributed.
Public LLM tokenizers	tiktoken `o200k_base` and `cl100k_base`; Hugging Face releases of Llama 3.1 8B, Qwen2.5-7B, Qwen3.6-27B, and Mistral-Nemo-Base-2407	MIT; Llama 3.1 Community License; Apache 2.0	Tokenizers only, for FLORES-200 token counts in Figure 2a and Tables H.2 and E.21; not redistributed.

I.2 Complete `101-lang` per-language results

Table I.3: Complete FLORES-200 mean tokens per sentence for all 204 languages under `101-lang` tokenizer training. Columns give vocabulary sizes; each language has paired UBE and BBPE rows. Means are rounded to three decimal places; the win/tie/loss counts in Table D.1 use unrounded values.

Language	Tokenizer	16K	32K	64K	128K	256K
Acehnese (Arabic)	UBE	**85.927**	**79.600**	**75.741**	**70.241**	**67.480**
	BBPE	86.626	79.774	75.897	70.374	67.620
Acehnese (Latin)	UBE	**60.001**	**56.095**	**52.427**	**49.271**	**45.328**
	BBPE	60.122	56.191	52.458	49.284	45.389
Afrikaans	UBE	**52.071**	**47.458**	**43.605**	**40.014**	**36.589**
	BBPE	52.299	47.583	43.651	40.024	36.590
Akan	UBE	**76.909**	**65.912**	**63.770**	**61.159**	**55.821**
	BBPE	77.001	65.955	63.786	61.182	55.850
Amharic	UBE	82.890	70.851	61.581	52.367	44.478
	BBPE	**81.970**	**69.995**	**60.261**	**51.053**	**43.655**
Arabic	UBE	**68.952**	**60.407**	**52.653**	**46.245**	**40.890**
	BBPE	69.872	60.600	52.753	46.296	40.905
Arabic (Latin)	UBE	**70.497**	**66.711**	**62.903**	59.892	**57.739**
	BBPE	70.679	66.736	62.952	**59.888**	57.740
Armenian	UBE	**91.947**	**78.250**	**64.792**	**54.347**	**46.934**
	BBPE	92.957	78.649	65.068	54.425	46.963
Assamese	UBE	**121.978**	107.690	**103.233**	**98.061**	**94.208**
	BBPE	122.269	**107.488**	103.420	98.105	**94.208**
Asturian	UBE	**52.166**	**47.708**	**44.491**	**41.299**	**38.654**
	BBPE	52.390	47.810	44.559	41.320	38.655
Awadhi	UBE	**89.941**	**87.054**	**85.436**	**84.299**	83.747
	BBPE	90.202	87.123	85.446	84.552	**83.743**
Ayacucho Quechua	UBE	**63.566**	**57.986**	**53.421**	**50.081**	**47.194**
	BBPE	63.680	58.379	53.437	50.100	47.196
Aymara	UBE	**63.904**	**60.158**	**56.439**	**53.707**	**51.015**
	BBPE	64.060	60.242	56.540	53.711	51.020
Azerbaijani	UBE	**69.600**	**60.773**	**52.889**	45.706	39.928
	BBPE	70.343	60.985	52.938	**45.656**	**39.887**
Balinese	UBE	**55.118**	**51.107**	**47.312**	**42.325**	**38.968**
	BBPE	55.354	51.183	47.353	42.358	38.996
Bambara	UBE	**72.784**	**60.037**	**58.458**	**56.308**	**53.743**
	BBPE	72.858	60.068	58.469	56.314	53.773
Banjar (Arabic)	UBE	**88.832**	**80.923**	**75.831**	**69.976**	**66.470**
	BBPE	89.302	81.011	75.842	69.978	66.504

Continued on next page

Table I.3 – continued from previous page

Language	Tokenizer	16K	32K	64K	128K	256K
Banjar (Latin)	UBE	**52.688**	**47.217**	**43.095**	**39.330**	**36.176**
	BBPE	52.796	47.273	43.111	39.360	**36.176**
Bashkir	UBE	**86.725**	**75.885**	**65.871**	**59.651**	**54.490**
	BBPE	87.014	76.205	65.915	59.676	54.504
Basque	UBE	**56.730**	**51.664**	**46.143**	**40.982**	**36.703**
	BBPE	56.903	51.779	46.189	41.031	36.706
Belarusian	UBE	**82.795**	**72.856**	**64.942**	**56.513**	**49.458**
	BBPE	83.050	73.020	65.007	56.563	49.490
Bemba	UBE	**65.952**	**60.496**	**57.039**	**54.178**	**50.822**
	BBPE	66.029	60.809	57.066	54.188	50.833
Bengali	UBE	**111.474**	104.137	**99.317**	95.335	**92.797**
	BBPE	111.689	**103.932**	99.507	**95.321**	92.844
Bhojpuri	UBE	89.326	**86.490**	**85.134**	**83.772**	83.337
	BBPE	**89.237**	86.524	85.141	84.063	**83.328**
Bosnian	UBE	**57.041**	**52.513**	**48.725**	**45.691**	**42.523**
	BBPE	57.147	52.595	48.825	45.699	42.525
Buginese	UBE	**59.070**	**54.659**	**51.357**	**48.553**	**45.399**
	BBPE	59.173	54.735	51.365	48.604	**45.399**
Bulgarian	UBE	**68.430**	**59.862**	**52.577**	**45.971**	**41.144**
	BBPE	69.152	60.066	52.661	46.033	**41.144**
Burmese	UBE	**139.676**	**125.853**	117.201	**111.051**	108.263
	BBPE	139.770	126.255	**117.161**	111.058	**108.256**
Cantonese	UBE	**48.922**	**43.513**	38.206	34.518	31.580
	BBPE	49.708	43.637	**37.856**	**34.116**	**31.221**
Catalan	UBE	**53.407**	**49.471**	**45.687**	**42.175**	**39.055**
	BBPE	53.569	49.581	45.742	42.190	39.064
Cebuano	UBE	**58.205**	**53.396**	**48.913**	**45.038**	**41.801**
	BBPE	58.409	53.467	48.954	45.063	41.805
Central Atlas Tamazight	UBE	**192.112**	**183.305**	**182.243**	**180.664**	**180.648**
	BBPE	269.200	269.166	269.123	269.102	203.206
Central Kanuri (Arabic)	UBE	**91.903**	**88.641**	**86.271**	**84.266**	**82.724**
	BBPE	91.996	88.654	86.276	84.267	82.729
Central Kanuri (Latin)	UBE	**67.723**	**63.558**	**60.579**	57.633	55.173
	BBPE	67.961	63.665	60.625	**57.618**	**55.159**
Central Kurdish	UBE	**93.146**	**86.587**	**79.674**	**70.182**	**61.794**
	BBPE	95.684	86.598	79.678	**70.182**	61.800
Chhattisgarhi	UBE	**88.629**	**85.262**	**84.000**	**83.224**	**82.785**
	BBPE	88.660	85.428	84.004	83.226	82.791
Chichewa	UBE	**61.234**	**55.689**	**51.690**	**46.685**	**42.545**
	BBPE	61.352	55.782	51.723	46.818	42.580
Chinese (Simplified)	UBE	**44.910**	39.359	35.015	31.574	28.664
	BBPE	44.920	**39.183**	**34.764**	**31.310**	**28.437**
Chinese (Traditional)	UBE	**49.919**	**44.313**	39.077	35.411	32.414
	BBPE	50.770	44.440	**38.741**	**34.998**	**32.013**
Chokwe	UBE	**57.408**	**52.807**	**49.780**	**47.661**	**45.042**
	BBPE	57.561	52.890	49.803	47.688	45.080
Crimean Tatar	UBE	**61.907**	**56.758**	**51.302**	**47.011**	**43.093**
	BBPE	62.021	56.830	51.457	47.042	43.098
Croatian	UBE	**55.903**	**51.588**	**47.775**	**44.815**	**41.678**
	BBPE	56.013	51.664	47.858	44.821	41.686
Czech	UBE	**59.761**	**53.849**	**47.946**	**42.535**	**37.554**
	BBPE	59.944	54.008	48.132	42.559	37.570
Danish	UBE	**51.159**	**46.673**	**42.433**	**38.464**	**35.242**
	BBPE	51.294	46.848	42.479	38.488	35.254

Continued on next page

Table I.3 – continued from previous page

Language	Tokenizer	16K	32K	64K	128K	256K
Dari	UBE	**66.958**	**57.616**	**49.862**	**43.035**	**37.850**
	BBPE	67.196	57.763	49.888	43.068	37.867
Dinka	UBE	**63.980**	**55.801**	**52.088**	**50.110**	**48.201**
	BBPE	64.145	55.947	52.095	50.118	48.219
Dutch	UBE	**53.293**	**48.189**	**43.544**	**39.364**	**35.599**
	BBPE	53.531	48.251	43.577	39.386	35.608
Dyula	UBE	**61.540**	**57.182**	**54.221**	**50.734**	**48.086**
	BBPE	61.633	57.230	54.233	50.757	48.093
Dzongkha	UBE	**311.873**	**309.494**	**265.685**	244.805	**178.047**
	BBPE	471.899	367.301	286.182	**235.167**	178.307
Egyptian Arabic	UBE	**67.151**	**58.900**	**51.562**	**45.346**	**40.486**
	BBPE	67.931	59.080	51.726	45.388	40.509
English	UBE	**34.671**	**31.068**	**28.695**	**27.158**	**26.271**
	BBPE	34.863	31.160	28.716	27.174	26.272
Esperanto	UBE	**51.917**	**46.989**	**42.915**	**39.470**	**36.386**
	BBPE	52.133	47.121	42.958	39.507	**36.386**
Estonian	UBE	**55.151**	**49.897**	**45.625**	**41.158**	**36.723**
	BBPE	55.427	50.057	45.656	41.181	36.735
Ewe	UBE	**76.564**	**68.753**	**65.405**	**61.973**	58.427
	BBPE	76.683	68.804	65.426	62.300	**58.424**
Faroese	UBE	**60.310**	**55.507**	**50.721**	**47.020**	43.441
	BBPE	60.559	55.625	50.798	47.048	**43.440**
Fijian	UBE	**63.968**	**59.466**	**55.543**	**52.398**	**49.574**
	BBPE	64.063	59.592	55.565	52.418	49.577
Finnish	UBE	**60.395**	**54.506**	**49.171**	**44.018**	**39.285**
	BBPE	60.664	54.867	49.185	44.038	39.304
Fon	UBE	**103.407**	**87.433**	**84.325**	**78.079**	**73.668**
	BBPE	103.517	87.460	84.337	78.090	73.673
French	UBE	**57.925**	**52.676**	**48.367**	**44.186**	**40.611**
	BBPE	58.072	52.753	48.390	44.231	40.626
Friulian	UBE	**57.823**	**54.107**	**50.199**	**47.690**	45.504
	BBPE	57.903	54.277	50.211	47.699	**45.503**
Galician	UBE	**52.798**	**47.705**	**43.805**	**39.214**	**35.696**
	BBPE	52.959	47.823	43.850	39.242	**35.696**
Ganda	UBE	**58.761**	**54.324**	**51.193**	**48.448**	**45.609**
	BBPE	58.819	54.426	51.207	48.500	45.613
Georgian	UBE	100.074	85.117	71.468	59.405	49.793
	BBPE	**97.688**	**82.745**	**70.135**	**58.003**	**48.937**
German	UBE	**58.225**	**52.011**	**47.137**	**42.310**	**37.913**
	BBPE	58.459	52.177	47.185	42.369	37.921
Greek	UBE	**90.479**	**77.161**	**66.125**	**56.125**	**48.226**
	BBPE	90.726	77.342	66.307	56.179	48.243
Guarani	UBE	**65.314**	**60.675**	**56.804**	**53.292**	50.566
	BBPE	65.500	60.755	56.818	53.304	**50.563**
Gujarati	UBE	**105.362**	99.345	93.423	**89.804**	86.962
	BBPE	105.835	**98.813**	**93.399**	89.846	**86.943**
Haitian Creole	UBE	**49.862**	**46.222**	**42.663**	**39.065**	35.862
	BBPE	49.914	46.280	42.720	39.073	**35.858**
Hausa	UBE	**58.594**	**52.477**	**47.054**	**42.788**	**39.174**
	BBPE	58.639	52.565	47.129	42.828	39.177
Hebrew	UBE	**64.113**	**55.574**	**48.778**	**42.666**	**37.625**
	BBPE	64.358	55.780	48.851	42.683	37.635
Hindi	UBE	**89.002**	**86.442**	**85.105**	**84.375**	**83.987**
	BBPE	89.078	86.494	85.114	84.376	83.991

Continued on next page

Table I.3 – continued from previous page

Language	Tokenizer	16K	32K	64K	128K	256K
Hungarian	UBE	**65.203**	**59.007**	**52.771**	**47.145**	**41.420**
	BBPE	65.443	59.067	52.841	47.198	41.448
Icelandic	UBE	**59.486**	**53.666**	**48.718**	**44.026**	**39.751**
	BBPE	59.710	53.771	48.808	44.042	39.766
Igbo	UBE	**70.128**	**66.196**	**59.599**	**53.670**	48.304
	BBPE	70.221	66.757	59.622	53.754	**48.291**
Ilocano	UBE	**62.476**	**57.350**	**52.608**	**50.105**	**47.943**
	BBPE	62.706	57.398	52.616	50.115	47.960
Indonesian	UBE	**51.095**	**44.751**	**39.268**	**34.455**	30.570
	BBPE	51.282	44.845	39.315	34.494	**30.569**
Iraqi Arabic	UBE	**67.438**	**59.284**	**51.823**	**45.466**	**40.291**
	BBPE	68.389	59.478	51.943	45.511	40.306
Irish	UBE	**65.688**	**59.298**	**53.939**	**48.930**	**43.929**
	BBPE	65.809	59.389	54.026	48.950	43.958
Italian	UBE	**56.121**	**51.148**	**46.933**	**42.953**	**39.047**
	BBPE	56.361	51.283	46.953	42.965	39.050
Japanese	UBE	**52.419**	46.273	40.783	36.098	32.338
	BBPE	52.668	**46.194**	**40.630**	**35.965**	**32.145**
Javanese	UBE	**52.634**	**47.536**	**43.192**	**38.537**	**34.464**
	BBPE	52.764	47.591	43.223	38.557	34.468
Kabiyè	UBE	**129.830**	**112.313**	**109.938**	**107.228**	**87.811**
	BBPE	129.873	112.336	109.958	107.237	87.816
Kabuverdianu	UBE	**52.273**	**48.034**	**45.555**	**43.003**	**40.566**
	BBPE	52.365	48.076	45.561	43.015	**40.566**
Kabyle	UBE	**68.512**	**64.688**	**62.457**	**59.657**	55.993
	BBPE	70.568	65.489	62.967	60.166	**55.835**
Kachin	UBE	**70.703**	**64.437**	**61.013**	**56.744**	**54.767**
	BBPE	70.808	64.487	61.032	56.751	**54.767**
Kamba	UBE	**60.790**	**55.084**	**52.022**	**49.361**	**47.375**
	BBPE	60.852	55.171	52.039	49.371	47.383
Kannada	UBE	121.886	**114.405**	**108.854**	**104.483**	**101.329**
	BBPE	**121.878**	115.094	109.013	104.505	**101.329**
Kashmiri (Arabic)	UBE	**92.493**	**86.303**	**79.726**	**73.982**	**68.068**
	BBPE	92.757	86.397	79.754	74.014	68.077
Kashmiri (Devanagari)	UBE	**91.532**	**88.371**	85.811	**84.222**	**82.965**
	BBPE	91.667	88.529	**85.792**	84.232	83.002
Kazakh	UBE	**74.634**	**64.325**	**54.984**	**46.899**	**39.974**
	BBPE	75.698	64.406	55.096	46.988	39.980
Khmer	UBE	145.965	**135.804**	**127.642**	**121.825**	118.121
	BBPE	**145.946**	136.047	127.747	121.860	**118.116**
Kikongo	UBE	**59.435**	**55.104**	**51.684**	**47.888**	**46.230**
	BBPE	59.529	55.150	51.697	**47.888**	46.231
Kikuyu	UBE	**87.112**	**74.997**	**72.792**	**70.577**	**68.090**
	BBPE	87.187	75.096	72.799	70.608	68.093
Kimbundu	UBE	**62.657**	**57.136**	**53.250**	**50.544**	**48.355**
	BBPE	62.847	57.305	53.274	50.551	48.368
Kinyarwanda	UBE	**62.855**	**58.812**	**54.961**	**50.840**	**46.585**
	BBPE	62.890	58.974	54.989	50.857	46.590
Korean	UBE	**44.092**	**38.633**	**34.112**	30.377	27.495
	BBPE	44.539	38.766	34.193	**30.375**	**27.481**
Kyrgyz	UBE	**73.696**	**65.523**	**57.050**	**49.417**	**42.621**
	BBPE	74.529	65.638	57.085	49.437	42.637
Lao	UBE	**115.340**	**99.961**	**87.657**	**79.355**	**72.767**
	BBPE	115.791	100.289	88.180	79.768	72.899

Continued on next page

Table I.3 – continued from previous page

Language	Tokenizer	16K	32K	64K	128K	256K
Latgalian	UBE	**62.662**	**57.833**	**53.175**	**49.880**	**46.937**
	BBPE	62.811	57.886	53.249	49.902	46.945
Latvian	UBE	**63.681**	**56.927**	**50.618**	**44.915**	**39.578**
	BBPE	63.818	57.020	50.708	44.976	39.583
Ligurian	UBE	**61.958**	**58.355**	**55.042**	**52.183**	**49.449**
	BBPE	62.092	58.480	55.069	52.199	**49.449**
Limburgish	UBE	**55.518**	**51.811**	**48.811**	**44.882**	**42.739**
	BBPE	55.757	51.863	48.829	44.896	42.743
Lingala	UBE	**55.718**	**51.343**	**48.868**	**46.344**	**43.278**
	BBPE	55.750	51.408	48.885	**46.344**	**43.278**
Lithuanian	UBE	**57.804**	**52.376**	**47.147**	**42.651**	**37.911**
	BBPE	58.010	52.514	47.202	42.671	37.916
Lombard	UBE	**61.415**	**58.139**	**54.782**	**51.868**	**49.361**
	BBPE	61.538	58.209	54.797	51.891	49.363
Luba-Kasai	UBE	**57.894**	**53.066**	**49.802**	**46.561**	44.423
	BBPE	57.994	53.200	49.825	46.584	**44.422**
Luo	UBE	**55.680**	**50.832**	**48.359**	**46.138**	**44.131**
	BBPE	55.740	51.055	48.375	46.149	44.134
Luxembourgish	UBE	**62.170**	**57.709**	**52.636**	**48.048**	**44.665**
	BBPE	62.391	57.796	52.678	48.112	44.670
Macedonian	UBE	**66.926**	**58.824**	**51.662**	**45.542**	**40.673**
	BBPE	67.254	58.984	51.717	45.597	40.676
Magahi	UBE	**89.901**	**87.107**	**85.821**	84.537	84.107
	BBPE	90.118	87.131	85.824	**84.531**	**84.106**
Maithili	UBE	**92.055**	**89.345**	**87.868**	85.727	84.039
	BBPE	92.080	89.391	87.870	**85.723**	**84.004**
Malay	UBE	**52.465**	**45.849**	**40.387**	**35.232**	31.275
	BBPE	52.641	45.912	40.434	35.286	**31.274**
Malayalam	UBE	132.650	126.980	120.760	115.976	**113.153**
	BBPE	**132.444**	**126.831**	**120.714**	**115.951**	113.167
Maltese	UBE	**71.571**	**65.837**	**62.167**	**57.209**	**53.027**
	BBPE	71.753	65.878	62.196	57.295	53.037
Manipuri	UBE	**124.214**	119.227	111.831	107.467	103.967
	BBPE	124.869	**119.007**	**111.738**	**106.202**	**103.147**
Maori	UBE	**67.069**	**61.654**	**56.111**	**51.477**	**46.518**
	BBPE	67.136	61.724	56.149	51.595	46.521
Marathi	UBE	**104.165**	**100.329**	**98.342**	96.261	95.370
	BBPE	104.264	100.625	98.368	**96.236**	**95.363**
Minangkabau (Arabic)	UBE	**92.803**	**83.785**	**78.774**	**72.147**	**68.735**
	BBPE	93.431	83.870	78.798	72.166	68.776
Minangkabau (Latin)	UBE	**54.254**	**50.000**	**45.832**	**42.494**	**39.906**
	BBPE	54.379	50.083	45.858	42.512	39.907
Mizo	UBE	**60.398**	**56.024**	**51.791**	**49.498**	46.544
	BBPE	60.598	56.080	51.802	49.518	**46.534**
Mongolian	UBE	**80.031**	**69.169**	**59.959**	**50.714**	**43.704**
	BBPE	80.340	69.547	60.095	50.801	43.726
Moroccan Arabic	UBE	**67.443**	**59.507**	**52.555**	**46.515**	**42.130**
	BBPE	68.267	59.719	52.646	46.535	42.162
Mossi	UBE	**65.381**	60.276	58.559	56.768	53.745
	BBPE	66.379	**59.651**	**57.896**	**56.103**	**53.059**
Najdi Arabic	UBE	**69.042**	**60.559**	**52.814**	**46.434**	**41.090**
	BBPE	69.959	60.746	52.919	46.484	41.105
Nepali	UBE	**96.657**	**93.235**	**91.076**	**89.527**	**88.408**
	BBPE	96.719	93.377	91.098	**89.527**	88.415

Continued on next page

Table I.3 – continued from previous page

Language	Tokenizer	16K	32K	64K	128K	256K
Nigerian Fulfulde	UBE	**54.955**	**50.787**	**46.216**	**43.593**	**41.066**
	BBPE	55.002	50.848	46.256	43.609	41.071
North Levantine Arabic	UBE	**64.101**	**56.463**	**49.510**	**43.852**	**39.072**
	BBPE	64.897	56.593	49.641	43.889	39.100
Northern Kurdish	UBE	**61.977**	**56.102**	**50.753**	**46.007**	**41.816**
	BBPE	62.231	56.165	50.822	46.066	41.831
Northern Sotho	UBE	**63.217**	**59.307**	**56.489**	**53.723**	**50.016**
	BBPE	63.522	59.399	56.526	53.741	50.020
Northern Uzbek	UBE	**62.423**	**57.289**	**51.243**	**45.904**	**40.856**
	BBPE	62.486	57.377	51.326	45.931	40.873
Norwegian Bokmål	UBE	**49.458**	**45.037**	**41.152**	**37.581**	**34.627**
	BBPE	49.628	45.210	41.177	37.622	34.633
Norwegian Nynorsk	UBE	**51.084**	**46.710**	**42.946**	**39.627**	**36.563**
	BBPE	51.270	46.811	43.007	39.681	36.568
Nuer	UBE	**109.780**	**93.545**	**81.559**	**79.516**	**72.085**
	BBPE	109.952	93.604	81.562	79.635	72.090
Occitan	UBE	**57.495**	**53.097**	**49.753**	**46.640**	**44.212**
	BBPE	57.662	53.154	49.778	46.650	44.223
Odia	UBE	188.713	**161.316**	135.328	**118.017**	109.633
	BBPE	**182.258**	164.863	**133.265**	**118.017**	**108.809**
Pangasinan	UBE	**48.309**	**44.354**	**41.247**	**38.685**	**36.988**
	BBPE	48.428	44.472	41.258	38.712	37.018
Papiamento	UBE	**53.667**	**49.781**	**46.553**	**43.911**	**41.106**
	BBPE	53.780	49.859	46.590	43.917	41.203
Plateau Malagasy	UBE	**65.920**	**60.520**	**55.479**	**50.181**	**46.322**
	BBPE	66.040	60.559	55.491	50.185	46.335
Polish	UBE	**65.246**	**58.423**	**52.398**	**46.618**	**40.905**
	BBPE	65.884	58.563	52.464	46.654	40.909
Portuguese	UBE	**52.184**	**47.341**	**43.180**	**38.896**	**35.369**
	BBPE	52.377	47.566	43.235	38.904	**35.369**
Punjabi	UBE	**106.408**	98.947	93.255	**88.974**	**86.336**
	BBPE	107.065	**98.726**	**93.199**	89.055	86.388
Romanian	UBE	**58.321**	**54.096**	**49.314**	**44.658**	**40.676**
	BBPE	58.457	54.194	49.376	44.672	40.681
Rundi	UBE	**63.596**	**59.159**	**55.311**	**51.373**	**47.621**
	BBPE	63.638	59.374	55.375	51.388	47.625
Russian	UBE	**71.353**	**61.758**	**53.181**	**46.146**	**40.539**
	BBPE	71.831	61.986	53.296	46.194	40.551
Samoan	UBE	**65.732**	**62.070**	**59.037**	**55.253**	**51.648**
	BBPE	65.797	62.167	59.042	55.294	51.650
Sango	UBE	**60.273**	**56.579**	**53.367**	**50.404**	**47.674**
	BBPE	60.302	56.607	53.372	50.420	47.680
Sanskrit	UBE	**99.349**	**95.658**	**93.779**	91.105	89.961
	BBPE	102.354	95.740	93.816	**91.095**	**89.953**
Santali	UBE	**237.326**	**222.944**	**212.692**	**211.712**	**207.379**
	BBPE	341.312	341.287	341.271	251.125	228.583
Sardinian	UBE	**60.539**	**55.834**	**52.812**	**49.657**	**46.819**
	BBPE	60.765	56.149	52.821	49.670	46.826
Scottish Gaelic	UBE	**69.524**	**64.659**	**60.289**	**55.730**	**50.936**
	BBPE	70.124	64.934	60.300	55.769	50.952
Serbian	UBE	**67.635**	**59.429**	**51.874**	**46.119**	**41.318**
	BBPE	67.838	59.612	51.923	46.154	**41.318**
Shan	UBE	236.768	221.375	218.032	**167.262**	151.673
	BBPE	**233.891**	**218.301**	**204.210**	170.984	**151.355**

Continued on next page

Table I.3 – continued from previous page

Language	Tokenizer	16K	32K	64K	128K	256K
Shona	UBE	**63.249**	**57.294**	**53.068**	**49.035**	**45.174**
	BBPE	63.323	57.509	53.125	49.199	45.180
Sicilian	UBE	**59.735**	**56.201**	**52.945**	**49.747**	**46.600**
	BBPE	60.358	56.648	53.291	50.107	46.944
Silesian	UBE	**71.144**	**62.284**	**58.068**	**54.112**	50.292
	BBPE	71.727	62.398	58.104	54.126	**50.288**
Sindhi	UBE	**88.438**	**75.398**	**64.024**	**56.548**	**49.625**
	BBPE	88.819	75.492	64.053	56.587	49.630
Sinhala	UBE	**110.344**	**102.037**	**95.809**	91.664	**88.621**
	BBPE	110.516	102.125	95.822	**91.593**	88.624
Slovak	UBE	**59.940**	**54.424**	**49.093**	**43.599**	**38.802**
	BBPE	60.335	54.685	49.178	43.622	38.815
Slovenian	UBE	**55.101**	**50.241**	**46.221**	**41.769**	**37.653**
	BBPE	55.371	50.386	46.295	41.807	37.662
Somali	UBE	**64.171**	**59.198**	**53.844**	**49.272**	**45.173**
	BBPE	64.639	59.266	53.864	49.355	45.187
South Azerbaijani	UBE	**71.596**	**64.139**	**57.975**	**52.104**	47.331
	BBPE	71.831	64.224	57.998	52.116	**47.329**
South Levantine Arabic	UBE	**63.855**	**56.180**	**49.159**	**43.426**	**38.605**
	BBPE	64.715	56.325	49.303	43.472	38.624
Southern Pashto	UBE	**71.269**	**61.397**	**53.629**	**47.345**	**42.423**
	BBPE	71.725	61.680	53.766	47.368	42.427
Southern Sotho	UBE	**64.929**	**60.443**	**56.135**	**53.151**	**50.300**
	BBPE	65.265	60.633	56.179	53.157	50.304
Spanish	UBE	**54.925**	**50.021**	**45.200**	**40.793**	**37.298**
	BBPE	55.205	50.161	45.262	40.805	37.300
Sundanese	UBE	**54.776**	**50.185**	**44.769**	**39.723**	**35.630**
	BBPE	55.007	50.275	44.827	39.742	35.632
Swahili	UBE	**55.131**	**50.284**	**45.010**	**39.962**	**35.767**
	BBPE	55.292	50.478	45.092	39.992	35.770
Swati	UBE	**63.442**	**58.921**	**55.185**	**51.379**	**47.642**
	BBPE	63.571	58.976	55.233	51.399	47.671
Swedish	UBE	**50.691**	**45.972**	**42.010**	**38.131**	**34.788**
	BBPE	50.818	46.118	42.037	38.165	34.792
Ta'izzi Arabic	UBE	**68.238**	**59.942**	**52.392**	**45.931**	**40.679**
	BBPE	69.195	60.128	52.545	45.988	40.692
Tagalog	UBE	**60.836**	**55.433**	**50.975**	**46.782**	**43.141**
	BBPE	60.985	55.543	51.022	46.794	43.153
Tajik	UBE	**82.025**	**72.294**	**64.915**	**57.818**	**50.596**
	BBPE	82.953	72.406	65.016	57.848	50.621
Tamasheq (Latin)	UBE	**64.325**	**60.727**	**57.668**	**55.734**	**51.245**
	BBPE	64.648	60.888	57.751	55.803	51.292
Tamasheq (Tifinagh)	UBE	**194.050**	**188.255**	**185.609**	**184.284**	**184.142**
	BBPE	271.147	270.655	270.571	270.528	204.566
Tamil	UBE	**130.214**	**122.921**	**118.113**	**115.209**	**113.794**
	BBPE	130.302	123.013	118.227	115.250	113.797
Tatar	UBE	**77.965**	**66.208**	**57.925**	**50.683**	**43.549**
	BBPE	78.247	66.621	57.987	50.709	43.575
Telugu	UBE	**114.800**	**110.878**	105.106	**100.440**	**98.118**
	BBPE	114.948	111.162	**105.057**	100.465	98.162
Thai	UBE	**75.640**	69.237	**64.534**	60.990	**58.107**
	BBPE	75.850	**69.173**	64.612	**60.969**	58.122
Tibetan	UBE	**287.065**	**286.737**	**247.171**	230.166	163.501
	BBPE	432.437	336.804	264.208	**214.252**	**161.215**

Continued on next page

Table I.3 – continued from previous page

Language	Tokenizer	16K	32K	64K	128K	256K
Tigrinya	UBE	**91.849**	79.221	**69.298**	61.681	54.403
	BBPE	92.910	**78.665**	69.399	**61.069**	**53.665**
Tok Pisin	UBE	**62.981**	**57.913**	**54.843**	**52.291**	**50.361**
	BBPE	63.056	57.952	54.853	52.299	50.365
Tosk Albanian	UBE	**61.181**	**56.013**	**51.111**	**46.273**	**41.842**
	BBPE	61.465	56.144	51.289	46.292	41.854
Tsonga	UBE	**65.786**	**61.653**	**58.231**	**55.727**	**52.549**
	BBPE	65.937	61.708	58.242	55.729	52.577
Tswana	UBE	**65.866**	**61.867**	**58.373**	**55.719**	52.893
	BBPE	66.321	62.021	58.438	55.723	**52.887**
Tumbuka	UBE	**74.959**	**68.298**	**64.038**	**58.219**	**54.144**
	BBPE	75.073	68.649	64.180	58.237	54.148
Tunisian Arabic	UBE	**66.258**	**58.247**	**51.588**	**45.781**	**41.058**
	BBPE	67.125	58.427	51.682	45.815	41.079
Turkish	UBE	**59.896**	**52.898**	**46.661**	**40.892**	**35.960**
	BBPE	60.139	52.972	46.795	40.961	35.966
Turkmen	UBE	**68.897**	**62.516**	**59.187**	56.246	**53.481**
	BBPE	69.116	62.595	59.215	**56.235**	53.517
Twi	UBE	**72.109**	**60.609**	**58.342**	**55.848**	**49.776**
	BBPE	72.292	60.657	58.359	55.874	49.794
Ukrainian	UBE	**72.472**	**63.801**	**56.567**	**49.195**	**43.529**
	BBPE	72.595	64.037	56.634	49.225	43.552
Umbundu	UBE	**59.733**	**55.394**	**52.952**	**50.009**	**47.377**
	BBPE	59.838	55.542	52.970	50.014	47.378
Urdu	UBE	**72.123**	**63.268**	**54.393**	**47.501**	**42.270**
	BBPE	72.404	63.449	54.484	47.551	42.296
Uyghur	UBE	**119.602**	**112.410**	**107.736**	**100.045**	**90.004**
	BBPE	120.675	112.438	107.739	100.048	90.006
Venetian	UBE	**53.993**	**50.199**	**46.913**	**43.812**	**41.147**
	BBPE	54.225	50.297	46.931	43.834	41.148
Vietnamese	UBE	**69.613**	55.407	**46.327**	39.947	36.715
	BBPE	69.764	**55.400**	46.381	**39.937**	**36.706**
Waray	UBE	**60.043**	**54.536**	**50.311**	**47.288**	**44.137**
	BBPE	60.254	54.592	50.323	47.311	44.139
Welsh	UBE	**66.409**	**61.948**	**58.707**	**54.242**	50.725
	BBPE	66.582	61.978	58.736	54.253	**50.723**
West Central Oromo	UBE	**69.103**	**64.926**	**60.678**	**57.090**	**53.208**
	BBPE	69.242	64.967	60.695	57.106	53.213
Western Persian	UBE	**69.852**	**60.530**	**52.618**	**45.601**	**40.308**
	BBPE	70.073	60.675	52.650	45.659	40.328
Wolof	UBE	**57.242**	**52.743**	**50.206**	**47.573**	**45.729**
	BBPE	57.316	52.815	50.213	47.591	45.736
Xhosa	UBE	**61.492**	**56.750**	**52.813**	**48.445**	**44.486**
	BBPE	61.587	56.796	52.856	48.456	44.491
Yiddish	UBE	**84.912**	**70.776**	**62.908**	**54.900**	**48.788**
	BBPE	85.573	71.280	63.101	54.911	48.797
Yoruba	UBE	**81.966**	**74.621**	70.871	**65.355**	61.404
	BBPE	83.391	75.041	**70.604**	65.383	**61.239**
Zulu	UBE	**65.370**	**60.117**	**55.212**	**50.784**	**45.533**
	BBPE	65.469	60.134	55.272	50.792	45.556

I.3 Per-language downstream benchmarks and held-out BPB

Tables I.4 and I.5 report per-language zero-shot accuracy for XNLI and XCOPA, and Table I.6 reports held-out BPB, under the primary `101-lang/32K` configuration (182M). Δ denotes UBE $-$ BBPE.

Section 3.2 summarizes held-out BPB differences by encoding-floor group; script-family aggregates can be recomputed directly from the per-language tables below.

Table I.4: Per-language XNLI zero-shot accuracy under the primary `101-lang/32K` 182M configuration (15 languages). Δ = UBE − BBPE.

Language	UBE	BBPE	Δ
ar	**33.6**	31.8	+1.8
bg	**33.4**	33.2	+0.2
de	**34.1**	33.4	+0.7
el	33.4	**33.7**	-0.3
en	40.4	**42.0**	-1.6
es	**34.0**	33.4	+0.6
fr	**35.2**	34.7	+0.5
hi	33.4	**33.5**	-0.1
ru	**35.9**	34.7	+1.1
sw	**35.0**	34.1	+0.8
th	33.6	**34.3**	-0.7
tr	**33.3**	33.1	+0.2
ur	33.3	**33.6**	-0.2
vi	33.1	**34.5**	-1.4
zh	34.3	**34.8**	-0.5
Mean	**34.4**	34.3	+0.1

Table I.5: Per-language XCOPA zero-shot accuracy under the primary `101-lang/32K` 182M configuration (11 languages). Δ = UBE − BBPE.

Language	UBE	BBPE	Δ
et	49.8	**50.8**	-1.0
ht	**52.4**	52.0	+0.4
id	52.2	**53.4**	-1.2
it	**49.6**	48.4	+1.2
qu	48.6	**50.0**	-1.4
sw	53.4	**54.6**	-1.2
ta	**54.0**	53.6	+0.4
th	55.4	**56.0**	-0.6
tr	**53.2**	52.8	+0.4
vi	**50.2**	50.0	+0.2
zh	**51.4**	50.2	+1.2
Mean	51.8	**52.0**	-0.1

Held-out BPB (101 languages).

Table I.6: Per-language held-out BPB for UBE and BBPE (`101-lang/32K`, 182M; 101 languages). UBE token-level perplexity is shown only as a scale reference; ΔBPB = UBE − BBPE.

Language	UBE BPB	BBPE BPB	ΔBPB	UBE PPL
af	1.9795	**1.9670**	+0.0125	52.31
am	1.0483	**1.0390**	+0.0093	9.50
ar	**1.2544**	1.2644	-0.0100	17.86
az	**1.9210**	1.9380	-0.0170	36.04
be	1.4363	**1.4010**	+0.0353	30.66
bg	1.1601	**1.1531**	+0.0070	21.12
bn	1.0159	**0.9728**	+0.0431	9.55
ca	1.6012	**1.5773**	+0.0239	25.30
ceb	**1.6769**	1.6802	-0.0033	24.38
co	1.4023	**1.3969**	+0.0054	12.53
cs	**1.9726**	1.9860	-0.0134	32.29

Table I.6 – continued from previous page

Language	UBE BPB	BBPE BPB	ΔBPB	UBE PPL
cy	**1.1069**	1.1240	-0.0171	10.97
da	1.4187	**1.3983**	+0.0204	16.50
de	1.5568	**1.5508**	+0.0060	21.65
el	**0.9531**	0.9781	-0.0250	9.39
en	1.0282	**1.0214**	+0.0068	14.10
eo	2.1549	**2.1384**	+0.0165	59.73
es	1.3764	**1.3663**	+0.0101	17.61
et	2.2668	**2.2282**	+0.0386	57.55
eu	1.7250	**1.6232**	+0.1018	22.09
fa	**1.2241**	1.2470	-0.0229	19.65
fi	1.9675	**1.9389**	+0.0286	32.96
fil	1.7348	**1.7185**	+0.0163	37.87
fr	1.4166	**1.4114**	+0.0052	17.89
fy	1.4440	**1.4229**	+0.0211	13.91
ga	1.5320	**1.4588**	+0.0732	17.52
gd	0.9800	**0.9611**	+0.0189	4.59
gl	1.6125	**1.6051**	+0.0074	25.43
gu	1.5372	**1.5355**	+0.0017	31.87
ha	2.1483	**2.1454**	+0.0029	57.05
haw	**2.0418**	2.0489	-0.0071	29.10
hi	0.7297	**0.7263**	+0.0034	6.25
hmn	1.8521	**1.8462**	+0.0059	21.54
ht	2.1021	**2.0850**	+0.0171	44.17
hu	**2.1818**	2.2069	-0.0251	41.04
hy	**1.0273**	1.2238	-0.1965	9.37
id	1.2899	**1.2777**	+0.0122	14.39
ig	**2.1886**	2.2380	-0.0494	45.18
is	2.2356	**2.2252**	+0.0104	55.78
it	**1.5506**	1.5512	-0.0006	23.09
iw	**1.7980**	1.7994	-0.0014	50.13
ja	0.9237	**0.9168**	+0.0069	8.65
jv	**1.8007**	1.8128	-0.0121	28.74
ka	1.2649	**1.0770**	+0.1879	42.92
kk	0.9030	**0.7750**	+0.1280	9.59
km	**1.3200**	1.4587	-0.1387	18.18
kn	**0.6466**	0.6689	-0.0223	4.12
ko	**0.8818**	**0.8818**	0.0000	10.05
ku	**1.3519**	1.4384	-0.0865	11.48
ky	**0.8469**	0.8660	-0.0191	8.08
la	1.4653	**1.4504**	+0.0149	18.31
lb	1.0129	**1.0088**	+0.0041	6.25
lo	**1.4836**	1.9145	-0.4309	35.47
lt	1.8119	**1.7687**	+0.0432	24.75
lv	**1.3041**	1.3562	-0.0521	9.76
mg	1.9600	**1.9434**	+0.0166	36.28
mi	**1.9730**	1.9811	-0.0081	31.00
mk	1.1655	**1.1293**	+0.0362	24.85
ml	**0.5524**	0.5549	-0.0025	3.47
mn	**1.6991**	1.7137	-0.0146	43.94
mr	0.9251	**0.9232**	+0.0019	10.53
ms	1.5119	**1.4916**	+0.0203	21.15
mt	**1.6258**	1.6412	-0.0154	10.10
my	1.4408	**1.0904**	+0.3504	34.47
ne	0.9534	**0.9511**	+0.0023	9.27
nl	1.5384	**1.5239**	+0.0145	21.90
no	**1.5625**	1.5628	-0.0003	22.33
ny	**1.9924**	2.0106	-0.0182	40.73
pa	1.7135	**1.5858**	+0.1277	52.47
pl	1.8785	**1.8636**	+0.0149	23.20
ps	1.8104	**1.7712**	+0.0392	73.08
pt	**1.2234**	1.2306	-0.0072	11.92
ro	1.9984	**1.9858**	+0.0126	49.56

Table I.6 – continued from previous page

Language	UBE BPB	BBPE BPB	ΔBPB	UBE PPL
ru	0.8143	**0.8056**	+0.0087	8.97
sd	0.8137	**0.8039**	+0.0098	5.42
si	1.5479	**1.2656**	+0.2823	33.97
sk	2.1511	**2.0940**	+0.0571	43.93
sl	**1.4659**	1.4815	-0.0156	13.61
sm	1.8296	**1.8198**	+0.0098	22.49
sn	1.9315	**1.9302**	+0.0013	33.36
so	**1.7567**	1.7942	-0.0375	26.43
sq	**2.7314**	2.7785	-0.0471	180.35
sr	1.4024	**1.3963**	+0.0061	35.68
st	**2.2992**	2.3017	-0.0025	55.37
su	1.8481	**1.8349**	+0.0132	35.17
sv	**1.6863**	1.7006	-0.0143	28.82
sw	**2.2022**	2.2039	-0.0017	58.59
ta	0.8556	**0.8134**	+0.0422	7.15
te	**0.9789**	1.0941	-0.1152	8.41
tg	**1.4043**	1.4377	-0.0334	34.27
th	0.4792	**0.4784**	+0.0008	4.85
tr	**1.3062**	1.3492	-0.0430	10.93
uk	0.9754	**0.9425**	+0.0329	11.30
ur	1.5161	**1.5038**	+0.0123	40.99
uz	**2.1720**	2.1786	-0.0066	48.10
vi	1.6018	**1.5877**	+0.0141	25.09
xh	1.9580	**1.9329**	+0.0251	33.52
yi	1.6290	**1.6012**	+0.0278	39.59
yo	2.6140	**2.5907**	+0.0233	90.06
zh	1.0238	**1.0117**	+0.0121	9.87
zu	1.7078	**1.6966**	+0.0112	25.30
Mean	1.5237	**1.5174**	+0.0063	

I.4 Per-language byte-level inputs

Table I.7: Figure 2b by language: 3-byte UTF-8 characters (% of characters), UBE and BBPE base symbols per sentence before BPE merging, and UBE's saving (%), sorted by saving. UBE is shorter in 202 of the 204 languages in FLORES-200 and ties in 2; the first 39, above the rule, are written mostly in 3-byte characters.

		Base symbols per sentence		
Language	3-byte (%)	UBE	BBPE	Saving (%)
Tibetan	98.165	**290.111**	433.819	33.126
Japanese	96.732	**110.699**	165.127	32.962
Dzongkha	96.341	**316.936**	472.451	32.917
Thai	94.651	**242.582**	360.531	32.715
Cantonese	92.883	**76.346**	113.076	32.483
Burmese	92.648	**310.627**	460.008	32.473
Chinese (Traditional)	92.694	**77.546**	114.821	32.463
Khmer	92.002	**290.713**	430.000	32.392
Chinese (Simplified)	89.535	**81.108**	119.371	32.054
Shan	88.739	**351.182**	516.283	31.979
Lao	88.659	**242.489**	356.432	31.968
Malayalam	87.860	**277.219**	406.871	31.865
Tamil	86.777	**284.475**	416.641	31.722
Kannada	85.966	**253.668**	370.929	31.613
Georgian	85.335	**263.596**	384.961	31.527
Sanskrit	85.158	**234.670**	342.596	31.502
Manipuri	84.493	**249.367**	363.567	31.411
Odia	84.031	**245.209**	357.173	31.347
Nepali	83.749	**230.427**	335.447	31.307
Marathi	83.596	**241.389**	351.295	31.286
Bengali	83.550	**234.059**	340.595	31.279
Telugu	83.490	**240.195**	349.482	31.271
Assamese	83.022	**227.774**	331.089	31.205
Santali	82.093	**251.538**	364.936	31.073
Sinhala	82.000	**235.742**	341.953	31.060
Gujarati	80.929	**226.039**	327.142	30.905
Maithili	79.572	**229.630**	331.382	30.705
Amharic	79.365	**154.445**	222.777	30.673
Punjabi	78.972	**235.971**	340.091	30.615
Awadhi	78.945	**226.644**	326.628	30.611
Magahi	78.931	**224.012**	322.828	30.610
Hindi	78.593	**231.294**	333.076	30.558
Central Atlas Tamazight	78.485	**208.052**	299.536	30.542
Chhattisgarhi	78.434	**223.812**	322.191	30.534
Tigrinya	77.783	**159.338**	229.050	30.435
Bhojpuri	77.731	**225.874**	324.658	30.427
Kashmiri (Devanagari)	75.552	**219.283**	313.648	30.086
Tamasheq (Tifinagh)	72.235	**210.949**	299.389	29.540
Korean	69.601	**110.560**	155.928	29.095
Vietnamese	10.990	**166.377**	181.429	8.296
Igbo	9.747	**146.150**	159.111	8.146
Yoruba	6.382	**158.852**	166.851	4.794
Kabyle	2.462	**134.824**	138.000	2.301
Samoan	2.027	**155.077**	158.136	1.935
Plateau Malagasy	1.376	**161.549**	163.738	1.337
Lombard	1.376	**149.400**	151.327	1.273
Mossi	1.369	**131.578**	133.265	1.266
Umbundu	1.153	**136.059**	137.594	1.115
Kinyarwanda	1.119	**145.769**	147.381	1.094
Guarani	1.063	**139.745**	141.140	0.989
Limburgish	0.976	**139.240**	140.568	0.945
Sicilian	0.989	**142.936**	144.291	0.939

Continued on next page

Table I.7 – continued from previous page

Language	3-byte (%)	Base symbols per sentence		Saving (%)
		UBE	BBPE	
Akan	0.939	**142.940**	144.170	0.853
Scottish Gaelic	0.849	**165.754**	167.119	0.817
Western Persian	1.404	**219.132**	220.851	0.778
West Central Oromo	0.765	**155.757**	156.940	0.754
Ligurian	0.764	**151.482**	152.585	0.723
Sardinian	0.646	**154.812**	155.791	0.628
Venetian	0.647	**137.075**	137.927	0.618
Ganda	0.614	**134.319**	135.138	0.606
Asturian	0.588	**138.636**	139.429	0.568
Maltese	0.579	**151.136**	151.973	0.551
Tamasheq (Latin)	0.543	**130.805**	131.473	0.508
Afrikaans	0.451	**138.994**	139.617	0.446
Fon	0.494	**164.313**	164.973	0.400
Lithuanian	0.335	**137.123**	137.553	0.313
Tsonga	0.303	**156.802**	157.277	0.302
Twi	0.324	**136.708**	137.120	0.301
Buginese	0.305	**141.483**	141.903	0.296
Occitan	0.297	**153.086**	153.528	0.288
French	0.291	**161.530**	161.983	0.280
Hausa	0.267	**139.657**	140.027	0.264
Nigerian Fulfulde	0.246	**123.362**	123.659	0.241
Rundi	0.208	**145.993**	146.296	0.207
Papiamento	0.195	**140.810**	141.078	0.190
Aymara	0.193	**139.932**	140.196	0.188
Kimbundu	0.188	**144.546**	144.817	0.187
Polish	0.179	**146.246**	146.493	0.169
Czech	0.159	**140.527**	140.726	0.142
Bosnian	0.140	**134.077**	134.260	0.136
Kabuverdianu	0.137	**132.203**	132.379	0.134
Friulian	0.136	**147.511**	147.705	0.131
Ayacucho Quechua	0.131	**140.297**	140.480	0.130
Basque	0.130	**138.426**	138.606	0.130
Kamba	0.133	**131.918**	132.087	0.128
Acehnese (Arabic)	0.225	**197.266**	197.514	0.126
Wolof	0.128	**130.475**	130.636	0.123
Slovak	0.133	**141.861**	142.034	0.122
Luo	0.121	**136.102**	136.266	0.120
Malay	0.119	**144.667**	144.839	0.119
Estonian	0.116	**131.748**	131.896	0.112
Bemba	0.112	**159.052**	159.230	0.112
Chichewa	0.111	**145.611**	145.773	0.111
Bulgarian	0.202	**247.332**	247.607	0.111
Romanian	0.116	**154.893**	155.064	0.110
Bashkir	0.203	**242.029**	242.295	0.110
Dyula	0.111	**139.745**	139.891	0.105
German	0.105	**154.344**	154.504	0.104
Ilocano	0.103	**157.993**	158.156	0.103
Chokwe	0.103	**137.976**	138.119	0.103
Macedonian	0.186	**247.962**	248.216	0.102
Faroese	0.104	**142.798**	142.937	0.097
Serbian	0.167	**234.955**	235.170	0.092
Norwegian Nynorsk	0.088	**134.715**	134.832	0.086
Somali	0.086	**150.011**	150.140	0.086
Javanese	0.083	**135.448**	135.560	0.083
Southern Sotho	0.083	**157.012**	157.141	0.082
Central Kanuri (Latin)	0.079	**144.735**	144.844	0.075
Hungarian	0.080	**150.663**	150.773	0.073
Banjar (Latin)	0.072	**136.671**	136.770	0.072
Ewe	0.078	**138.248**	138.347	0.071

Continued on next page

Table I.7 – continued from previous page

Language	3-byte (%)	Base symbols per sentence		Saving (%)
		UBE	BBPE	
Zulu	0.071	**146.736**	146.840	0.071
Swahili	0.070	**136.572**	136.668	0.070
Latvian	0.069	**144.828**	144.919	0.063
Acehnese (Latin)	0.058	**142.662**	142.744	0.057
Belarusian	0.100	**267.537**	267.683	0.055
Esperanto	0.056	**132.084**	132.156	0.055
Tswana	0.050	**161.231**	161.311	0.050
Croatian	0.049	**130.537**	130.599	0.048
Luxembourgish	0.048	**150.003**	150.073	0.047
Irish	0.049	**160.243**	160.317	0.046
Icelandic	0.050	**141.684**	141.748	0.045
Ukrainian	0.081	**242.863**	242.970	0.044
Northern Uzbek	0.044	**147.636**	147.702	0.044
Russian	0.078	**257.145**	257.255	0.043
Fijian	0.039	**153.422**	153.482	0.039
Kikuyu	0.042	**168.544**	168.608	0.038
English	0.037	**130.481**	130.530	0.037
Finnish	0.039	**144.035**	144.088	0.037
Kyrgyz	0.068	**247.142**	247.233	0.037
Minangkabau (Latin)	0.036	**139.568**	139.619	0.036
Dari	0.063	**212.392**	212.467	0.035
Xhosa	0.033	**137.547**	137.593	0.033
Tok Pisin	0.032	**167.088**	167.141	0.032
Turkish	0.035	**146.096**	146.142	0.032
Central Kanuri (Arabic)	0.056	**208.804**	208.869	0.031
Turkmen	0.032	**152.726**	152.771	0.029
Banjar (Arabic)	0.053	**220.793**	220.858	0.029
Minangkabau (Arabic)	0.052	**228.610**	228.675	0.029
Latgalian	0.029	**136.215**	136.253	0.028
Italian	0.028	**155.352**	155.394	0.027
Silesian	0.028	**143.768**	143.805	0.026
Urdu	0.046	**228.893**	228.953	0.026
Mizo	0.025	**144.794**	144.831	0.025
Crimean Tatar	0.028	**145.581**	145.618	0.025
Luba-Kasai	0.024	**141.370**	141.404	0.024
Kazakh	0.041	**246.962**	247.017	0.022
Haitian Creole	0.021	**123.166**	123.192	0.021
Sango	0.020	**145.608**	145.635	0.019
Indonesian	0.019	**140.731**	140.758	0.019
Waray	0.018	**162.848**	162.876	0.018
Swedish	0.018	**135.737**	135.761	0.017
Norwegian Bokmål	0.017	**133.406**	133.429	0.017
Dinka	0.019	**124.844**	124.865	0.017
Slovenian	0.017	**132.398**	132.420	0.016
Mongolian	0.028	**249.001**	249.040	0.015
Tagalog	0.014	**164.281**	164.304	0.014
Tajik	0.025	**262.334**	262.371	0.014
Danish	0.014	**136.333**	136.352	0.014
Sindhi	0.023	**209.341**	209.369	0.013
South Levantine Arabic	0.024	**194.493**	194.519	0.013
Lingala	0.013	**140.681**	140.699	0.013
Arabic (Latin)	0.012	**151.981**	152.000	0.012
Kashmiri (Arabic)	0.022	**225.446**	225.473	0.012
Portuguese	0.012	**145.683**	145.700	0.012
Kabiyè	0.012	**178.555**	178.572	0.009
Tatar	0.016	**241.837**	241.858	0.009
Balinese	0.008	**144.211**	144.223	0.008
Cebuano	0.008	**156.844**	156.857	0.008
Uyghur	0.015	**258.111**	258.131	0.008

Continued on next page

Table I.7 – continued from previous page

Language	3-byte (%)	Base symbols per sentence		Saving (%)
		UBE	BBPE	
Tumbuka	0.006	**172.764**	172.775	0.006
Pangasinan	0.006	**128.394**	128.402	0.006
Southern Pashto	0.011	**212.294**	212.307	0.006
Kikongo	0.006	**148.288**	148.296	0.006
Northern Sotho	0.006	**153.067**	153.076	0.006
South Azerbaijani	0.010	**212.487**	212.499	0.006
Swati	0.005	**146.984**	146.992	0.005
Yiddish	0.009	**253.591**	253.604	0.005
Azerbaijani	0.006	**164.561**	164.569	0.005
Armenian	0.008	**266.447**	266.458	0.004
Sundanese	0.004	**136.594**	136.599	0.004
Welsh	0.004	**140.111**	140.116	0.004
Greek	0.006	**283.805**	283.815	0.003
Northern Kurdish	0.004	**143.010**	143.015	0.003
Galician	0.003	**147.957**	147.961	0.003
Bambara	0.003	**135.931**	135.935	0.003
Dutch	0.003	**145.975**	145.979	0.003
Tosk Albanian	0.003	**156.623**	156.626	0.003
Iraqi Arabic	0.004	**203.959**	203.964	0.002
Ta'izzi Arabic	0.004	**206.798**	206.803	0.002
Arabic	0.004	**208.753**	208.758	0.002
Najdi Arabic	0.004	**209.029**	209.034	0.002
Nuer	0.003	**172.701**	172.705	0.002
North Levantine Arabic	0.004	**193.583**	193.587	0.002
Shona	0.002	**145.869**	145.872	0.002
Tunisian Arabic	0.004	**201.106**	201.110	0.002
Egyptian Arabic	0.004	**204.047**	204.051	0.002
Kachin	0.002	**167.405**	167.408	0.002
Moroccan Arabic	0.003	**203.695**	203.698	0.001
Hebrew	0.002	**180.908**	180.910	0.001
Maori	0.001	**151.976**	151.977	0.001
Spanish	0.001	**158.002**	158.003	0.001
Catalan	0.000	**147.049**	**147.049**	0.000
Central Kurdish	0.000	**233.564**	**233.564**	0.000

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the tokenizer-level contribution, distinguish intrinsic token-budget results from LM-quality results, and limit the scope to the part of token-budget disparity that the byte encoding imposes.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The conclusion summarizes the substrate-level scope, and Appendix H.4 discusses evaluation choices, low-coverage regimes, comparison scope, routing/streaming limits, and matched-pipeline filtering.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper's only formal result is the exact round-trip proposition in Appendix A.4, and that appendix states the decoding assumptions and gives a complete proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 3 and Appendix Sections B.1, B.2, E, and I.2 specify datasets, splits, metrics, tokenizer settings, matched LM protocols, statistical analyses, and comparisons with alternative tokenizers.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The UBE tokenizer implementation, the Unicode fidelity audit, and the MYTE-UBE transport and parity-verification code are available at https://github.com/Samsung/UBE; Appendix G states what the release covers. Tokenizer files, training corpora, and the LM training infrastructure are not part of the release.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The core paper reports evaluation metrics, dataset splits, and LM/tokenizer regimes. The appendix gives the training pipeline, hyperparameters, corpus token accounting, and per-language results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports paired language-bootstrap confidence intervals for all nine core Gini reductions, signed-rank tests with Holm and Benjamini–Hochberg multiple-comparison corrections, a multi-seed LM replication, and confidence intervals for prompt-processing and code-heavy-input measurements; Appendix Table E.1 gives the full statistics.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix G.2 specifies the 8× NVIDIA A100 80 GB setup, configured LM step counts, approximate per-run wall-clock/GPU-hour budgets, total GPU-hours after multiplying by run counts, plus CPU-only tokenizer and auxiliary-analysis costs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses public text corpora and released research code, and we reviewed the NeurIPS Code of Ethics and the licenses and release constraints for the assets used.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses the narrow fairness scope in the conclusion and Appendix H.4 covers both the intended multilingual-access benefits and plausible misuse risks.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The code release contains tokenizer and verification code, not pretrained LMs or training corpora.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Appendix I.1 lists external datasets and software assets with access points, license names, and usage constraints.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Appendix G.1 documents the tokenizer artifacts used in this work, including their use, training data, evaluation, limitations, and license. Documentation accompanies the code release described in Appendix G.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects, so no IRB-style approval was required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The research does not use LLMs as an important or non-standard component of the core method; released LLM tokenizers are evaluation baselines rather than a method-development tool.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.