WHOSE TOKENIZER? LANGUAGE-SPECIFIC EFFECTS OF TOKENIZER CHOICE IN MULTILINGUAL LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tokenizer choice affects multilingual language modeling, but vocabulary capacity is finite and vocabulary size is often constrained: improving representation for some languages often comes at the expense of others. We therefore ask whether tokenizer choice matters equally across languages. To this end, we train 123 language models spanning 54 tokenizers. In the main comparison, architecture, training corpus, training-token budget, and optimization are held fixed, so the models differ only in their tokenizer. We measure every outcome per language. Tokenizer choice matters more for languages with less language-model training data: across the 54 tokenizers, the standard deviation of a language's bits-per-byte (BPB) increases as its model training-data share decreases (Spearman $\rho = -0.52$ over the 31 trained languages and -0.69 over the 28 written with word boundaries). Leaving a language out of tokenizer training raises its BPB in every language we study, and the penalty tends to be larger for languages with less language-model training data. Giving lower-resource languages a larger share of tokenizer-training data, however, does not unconditionally help those languages: both equal weighting and an allocation inverted against their language model training data share increase their BPB, particularly when language-model training repeats data. Finally, which intrinsic tokenizer properties are associated with better BPB differs across languages, providing further evidence that what makes a good tokenizer depends on the language. Over the 31 languages, we achieve 0.89 accuracy on pairwise rankings of trained models' mean BPB using just tokenizer intrinsic metrics, suggesting a practical strategy for screening tokenizer candidates before training language models. All resources will be released upon de-anonymization.

1 INTRODUCTION

Multilingual language models serve dozens of languages and scripts with one tokenizer that is fixed before training. How that tokenizer segments each language has practical consequences: tokenization can create large differences in API costs across languages (Ahia et al., 2023), and prior work has raised concerns that some languages are systematically disadvantaged by current tokenizers (Petrov et al., 2023; Arnett et al., 2026). Controlled studies further show that tokenizer choice can substantially affect downstream model behavior when the rest of language-model training is held fixed (Ali et al., 2024; Singh & Strouse, 2024; Altıntaş et al., 2026; Lesci et al., 2025). These findings have motivated a range of modifications intended to improve multilingual tokenization (Foroutan et al., 2026; Land & Arnett, 2025; Asgari et al., 2025; Jabbar, 2024).

Whether these modifications help, and for which languages, is hard to establish from existing evidence because most results are aggregated too coarsely. Vocabulary capacity, measured in the number of unique tokens it includes, is shared and finite in a multilingual tokenizer: a change that improves the representation of one language often reduces the representation available to others. An outcome averaged over languages can therefore change little in aggregate while individual languages have outcomes change in opposite directions. The question of whether a tokenizer is a good choice must be posed for each language separately, and the literature lacks an account at that level: how much tokenizer choice changes a single language's performance, which languages are affected most,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

whether a choice that helps one language also helps others, and which tokenizer properties account for the differences.

Our work closes this gap by first training 123 language models across 54 tokenizers. The main comparison holds architecture, training corpus, training-token budget, and optimization fixed, isolating tokenizer effects on language modeling performance. The 54 tokenizers vary along four design axes: pretokenization, tokenization algorithm, vocabulary encoding, and tokenizer-training data. We measure every outcome separately for each of the 31 languages of the language-model training mixture, and perform careful statistical analyses of the observed trends. With these analyses, we are able to answer four research questions (§ 3 to 6), each motivated by the result of the previous one. Their main findings are summarized below.

- **Tokenizer choice affects every language.** Changing the tokenizer changes every language's bits-per-byte (BPB) by more than the variation across model seeds; tokenizers do not rank consistently across languages (mean pairwise Kendall $\tau = 0.418$ between the per-language rankings, §3).
- **Languages with less language-model training data are the most sensitive to tokenizer choice.** The *spread* of a language's BPB across the 54 tokenizers rises as its share of the language-model training data falls (Spearman $\rho = -0.52$); excluding a language from tokenizer training likewise raises BPB more as its share of the language-model training data falls (§4).
- **Reallocating tokenizer-training data toward those languages *raises* their BPB.** Equal weighting and an allocation inverse to the language-model training shares both raise the BPB of the lower-resource languages (more so when language-model training repeats data), suggesting additional vocabulary for underrepresented languages is not unconditionally helpful (§5).
- **Intrinsic metrics measured per language can screen tokenizers without training a language model.** Which intrinsic tokenizer metrics are associated with lower BPB differs by language; measured per language, nine metrics order held-out tokenizer pairs by the trained models' mean BPB with 0.89 accuracy (§6).

From our results, we conclude that tokenizer selection is most consequential for the low-resource languages, that per-language intrinsic metrics can screen choices before any model training, and that tailoring to low-resource languages costs the high-resource languages little. Importantly, tailoring should be done by selection and coverage, not necessarily by tokenizer training data allocation. We derive practical and actionable recommendations for multilingual tokenizer evaluation and selection (§7).

Related studies. Prior works have investigated how tokenizer choice changes downstream language modeling performance, spanning arithmetic (Singh & Strouse, 2024), code (Dagan et al., 2024; Meister, 2026) and multilingual and general benchmarks (Ali et al., 2024; Altıntaş et al., 2026; Rust et al., 2021). Several do this in a controlled fashion: Ali et al. (2024) vary the algorithm and vocabulary size, Goldman et al. (2024) vary the amount of tokenizer-training data, and Altıntaş et al. (2026) compare 14 off-the-shelf tokenizers. While they establish the importance of tokenizer choice, a unified account of when and how much it matters—particularly on a per-language basis—is missing. Each evaluates English and at most four other languages, and they do not cover the breadth of training algorithms, pretokenization and normalization strategies, text encoding and data composition in an isolated fashion. In parallel, mechanisms behind these differences have been studied through intrinsic metrics that could be computed from the tokenizer alone without model training. Gallé (2019) and Goldman et al. (2024) find that compression rate correlates with downstream performance, whereas Schmidt et al. (2024), Lester et al. (2024), and Lotz et al. (2025) show that compression rate alone is an incomplete predictor of downstream model quality. Alternatives have been proposed with varying degrees of success, but the utility of these metrics on a per-language basis has not been assessed.

2 Experimental Setup

2.1 Tokenizer and Model Training

Tokenizers. We compare 54 tokenizers covering a diverse set of practices established or proposed for multilingual tokenization. We train 52 of them on mixtures of the same multilingual corpus. The other two are off-the-shelf tokenizers from the open-source LLMs Mistral-Nemo and LLaMA-3, with comparable vocabulary sizes to our panel. We outline the tokenizer design axes that we vary below, with more detail in §B.1.

- *Training algorithm.* We consider BPE (Sennrich et al., 2016), UnigramLM (Kudo, 2018), MinGram (Land, 2026), SuperBPE (Liu et al., 2025), parity-aware BPE (Foroutan et al., 2026), and combinations of the latter two.

- *Pretokenization and normalization.* Pretokenization determines which consecutive characters can form a token and how scripts, numbers, punctuation, and whitespace are handled (all detailed in §B.2). We also vary normalization across NFC, NFD, and no Unicode normalization.
- *Text encoding.* Most tokenizers operate on raw UTF-8 bytes; three instead use an encoding based on Unicode script classes (Land & Arnett, 2025), proposed to reduce the byte-level penalty that scripts used by low-resource languages incur.
- *Training data.* Our baseline *balanced* tokenizer training dataset consists of a balanced 10 GB mixture across domains. Concretely, its sampling weights are 35% English (FineWeb-Edu (Penedo et al., 2024)), 30% multilingual across 30 languages (quality-filtered FineWeb2 (Penedo et al., 2025), filtered per Apertus 1.0 (Hernández-Cano et al., 2026)), 15% math (FineMath (Lozhkov et al., 2024)), and 15% code (StarCoderData (Li et al., 2023)). The 31 natural languages span 11 writing systems and 11 language families. For six of the tokenizers, the multilingual pool comes from a sample of FineWeb2's full language pool. We perform targeted changes to this dataset composition, ablating language coverage and per-language sampling weights, following the range of practices used to build tokenizers for existing multilingual LLMs.

Each of these four axes has been proposed as a determinant of multilingual quality: the algorithm and the pretokenization determine which byte sequences can become vocabulary entries, the text encoding determines the per-character byte cost of each script, and the training data determines which languages' byte sequences are frequent enough to enter the vocabulary. Three tokenizers have known construction defects that are common artifacts of tokenizer training (a missing part of the byte-level initial alphabet, or a dropped lone space). We analyze the impact of the byte-alphabet defect in §D.

Language models. For each tokenizer, we train a corresponding 1.27 B-parameter language model, fixing architecture, training data, and optimization across all runs following the nanochat d24 configuration (Karpathy, 2025). We follow nanochat's scaling law recommendations, using 9.2 B tokens; §C shows that model rankings are robust to this design choice. We use the same corpora as for tokenizer training, following a composition close to that of the *balanced* mixture. Unless otherwise noted for ablations, the training mixture is held constant. Throughout, a language's *share of the language-model training data* (realized byte share) often differs from its *share of the tokenizer-training data* above. Because the number of training steps is held approximately constant, the amount of underlying text seen by each model may differ, as different tokenizers will have different compression rates. We refer to the underlying text seen by the model as its *training bytes*. Details on model architecture, optimization and training data are provided in § B.3 and B.4. To check whether tokenizer rankings hold beyond our main scale, a subset of tokenizers is also trained at three smaller scales: d8, d12, d16 (Table 3). We include these models only in the ranking analysis in §6.

2.2 Evaluation

Per-language outcomes. Our main outcome is per-language bits-per-byte (BPB) (Gao et al., 2021), a tokenizer-agnostic evaluation for language-modeling performance. Computation details are given in §B.5. We use FLORES+ (NLLB Team et al., 2022) as our main evaluation set, a corpus of the same sentences translated into many languages. The main analyses use the 31 languages of the language-model training mixture. Results using the other 183 languages in FLORES+ are provided in §F.2. For statements about a single language, we use that language's FLORES+ BPB. As secondary downstream outcomes, we compute per-language accuracy on MultiBLiMP (Jumelet et al., 2026), a multilingual minimal-pairs grammaticality benchmark covering 24 of the 31 languages, and on XCOPA (Ponti et al., 2020), a two-alternative causal-commonsense benchmark, covering 7 of the 31 languages (§G).

Aggregate outcomes. For comparisons that require a single outcome per tokenizer, we report two aggregate measures. The *31-language FLORES+ mean* is the mean of the run's FLORES+ BPB over the 31 languages of the language-model training mixture. The *Validation BPB* is BPB on held-out text from the language-model training mixture. Thus, the 31-language FLORES+ mean weights all 31 languages equally and evaluates on text outside the training distribution, whereas Validation BPB reflects the unequal language proportions of the language-model training distribution.

Intrinsic tokenizer metrics. Separately, we compute nine intrinsic tokenizer metrics with the TokEval library (Meister, 2026). These metrics serve as explanatory and predictive variables in §6. The metrics require no language-model training, but all the metrics we consider are computed with respect to a (user-defined) text. Unless otherwise stated, we therefore consider each metric on a per-language basis, i.e., computed on text from a specified language. §F.1 defines the metrics, and Fig. 3 summarizes them together with their measured within-language directions.

2.3 ANALYSIS

This section describes the analysis tools we employ throughout the work. In short, we use descriptive statistics and rank correlations to characterize how tokenizer effects vary across languages. We use linear mixed-effects models that account for differences in baseline performance across languages for the analyses relating intrinsic tokenizer properties to downstream outcomes. Finally, for the intrinsic metric predictive ranking analyses in §6, we use logistic pairwise rankers.

Associations between tokenizer metrics and downstream outcomes. Several analyses ask how a tokenizer metric m is associated with a downstream outcome across languages and tokenizers. Absolute BPB differs substantially across languages, with between-language differences much larger than the tokenizer-induced differences within any one language (§3). We therefore use a linear mixed-effects model that accounts for each language's baseline BPB level. For example, when FLORES+ BPB of tokenizer t on language ℓ is the outcome, we fit:

$$\text{BPB}_{t,\ell} = \beta_0 + \beta_m \, z(m_{t,\ell}) + b_\ell + \varepsilon_{t,\ell}, \qquad b_\ell \sim \mathcal{N}(0, \tau^2), \quad \varepsilon_{t,\ell} \sim \mathcal{N}(0, \sigma^2), \tag{1}$$

where $z(m_{t,\ell})$ standardizes the metric using the mean and standard deviation computed over all (t, ℓ) pairs in the analysis sample. The random intercept b_ℓ accounts for language-specific BPB baseline. After accounting for baseline differences, β_m gives the change in BPB associated with a one-standard-deviation increase in m. To test whether this relationship varies by resource level, writing system, or morphological type, we add the corresponding interaction terms.

Predictive ranking. For the analysis in §6, we train logistic pairwise rankers that predict, from the difference of two tokenizers' intrinsic-metric vectors, which of the two trained models has the lower BPB (implementation details in §F.4). To account for dependence among closely related tokenizers, we use leave-one-family-out cross-validation: each fold holds out an entire tokenizer family, so a tokenizer is never ranked by a model trained on another member of its own family. We evaluate pairwise accuracy, the fraction of scored tokenizer pairs whose predicted ordering matches their ordering by the trained models' BPB, over every pair with at least one tokenizer in the held-out family.

Statistical protocol. The standard errors used in significance testing come from measuring variation over seed replicates of models for a fixed tokenizer. We refer to the size of a change across seeds as the *seed variation*. For analyses of individual languages, we call a difference between two runs *larger than seed variation* when its magnitude exceeds the pooled seed standard deviation for a single language's FLORES+ BPB. For comparisons on Validation BPB and the 31-language FLORES+ mean, we use a stricter reference threshold: we call a difference *larger than run-to-run variation* when its magnitude is approximately outside the 95% range expected from two runs that differ only in random seed, i.e., when it is at least ∼twice the estimated standard deviation of the difference between two such runs. Exact protocol and values can be found in §B.5. All p-values use Benjamini–Hochberg false-discovery-rate correction within the relevant set of tests. In plots and tables, *, **, and *** indicates $p < 0.05$, $p < 0.01$ and $p < 0.001$, respectively.

We primarily report results per tokenizer, but for the sake of robustness, additionally assess results on a per-tokenizer-family basis. Concretely, results from the 54 tokenizers are not fully independent: some tokenizers differ only in a single training hyperparameter or preset. This breaks independence assumptions in several of the analysis methods we use. We therefore additionally present results when grouping tokenizers that differ only in such minor hyperparameters into the same *family*, and we rely primarily on cluster-robust standard errors, clustered at the tokenizer-family level. Full details of the family stratification are given in §B.1 and pooled standard error computation details in §B.5.

3 HOW MUCH DOES TOKENIZER CHOICE MATTER ACROSS LANGUAGES? (RQ1)

Finding 1. Changing the tokenizer changes every language's FLORES+ BPB, and MultiBLiMP accuracy for 21 of the 24 covered languages, by more than that language's seed variation, and the 54 tokenizers rank differently across languages; a tokenizer chosen on an aggregate outcome is therefore a compromise across languages rather than the best tokenizer for each of them.

Language identity accounts for nearly all of the variance in the per-(tokenizer, language) BPB measurements, so the tokenizer effect is visible only within a language. Language identity accounts for 99.4% of the variance in per-(tokenizer, language) BPB, so a tokenizer changes only the within-language remainder. That remainder is largest for Chinese, with a BPB standard deviation

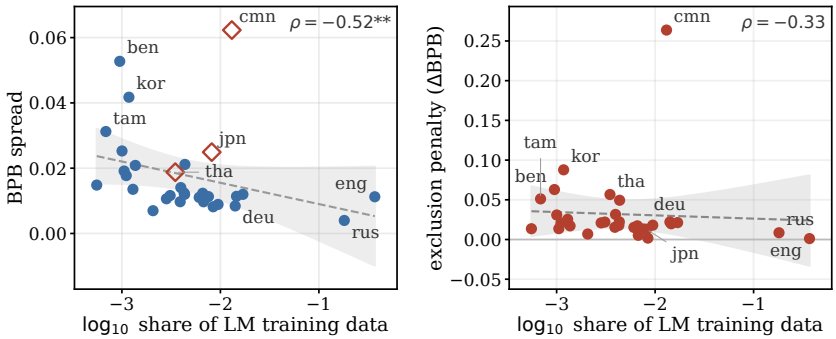

Figure 1: **Left**: the standard deviation of a language's FLORES+ BPB across the 54 tokenizers (BPB spread) vs. the language's log share of the language-model training data (Spearman $\rho = -0.52$, $p = 0.003$). Open diamonds mark the three languages written without word boundaries (Chinese, Japanese, and Thai). **Right**: the exclusion penalty (defined in the text) against the same axis (Spearman $\rho = -0.33$, $p = 0.068$). The horizontal line marks a penalty of zero. Dashed lines are ordinary-least-squares fits, and the grey regions are 95% confidence regions for the fitted mean.

across the 54 tokenizers of 0.062 and a range of 0.305, followed by Bengali and Korean. The result holds under two alternative measures of the spread (§C). Every analysis that follows therefore compares tokenizers within a language.

The spread across tokenizers is greater than the seed variation for every language on FLORES+ BPB and for all but three on MultiBLiMP, so there is a measurable tokenizer effect for every language, not only the low-resource ones. The standard deviation in per-language FLORES+ BPB across the 54 tokenizers is greater than the language's seed variation for all 31 languages; on MultiBLiMP accuracy, this seed variation is exceeded for 21 of the 24 languages that MultiBLiMP covers. For 16 of the 24 languages, however, the pooled seed standard deviation does not exceed the sampling uncertainty from the benchmark's finite number of minimal pairs (the binomial standard error), so for these languages run-to-run variation cannot be measured more finely than the benchmark's own sampling noise (§G). Fig. 4 visualizes this across languages. On XCOPA, accuracy stays near chance for every covered language, so we do not consider these results in our conclusions (§G).

Per-language tokenizer rankings agree only partially across languages, so no single tokenizer is best for all of them. The mean agreement between the per-language tokenizer rankings (the off-diagonal Kendall τ between two orderings, 1 for identical and 0 for unrelated) is 0.418. The ranking agreement is lowest for Ukrainian (0.099). The general disagreement is not specific to the FLORES+ text, although Ukrainian's extreme value is, suggesting there may be some artifacts in this particular text. At this level of agreement, 16 of the 54 tokenizers rank first for at least one language, i.e., none ranks first for every language. The consequence of this disagreement for a single shared tokenizer is discussed next.

4 Which Languages Are Most Tokenizer-Sensitive? (RQ2)

Finding 2. Lower-resource languages, those with a smaller share of the language-model training data, are more sensitive to tokenizer choice under two measures: First, excluding a language from tokenizer training raises its BPB for all 31 languages, and more so when it has a smaller share in the language-model training data. Second, each language's BPB spread across 54 tokenizers also falls as its share of the language-model training data rises. We find writing system and resource-level effects are confounded (§C).

A language's BPB varies less across tokenizers as its share of the language-model training data rises, so the languages with the least language-model training data are most sensitive to tokenizer choice. Fig. 1 (left) shows tokenizer performance spread (standard deviation of FLORES+ BPB across the 54 tokenizers) as a function of log language-model training data share. The two are negatively correlated (Spearman $\rho = -0.52$), more strongly among the 28 languages with word boundaries ($\rho = -0.69$); both significant. This is robust to how spread is aggregated: computing it on one mean per tokenizer family gives approximately the same (significant) correlation (-0.54). Russian and German, two of the highest-resource languages in the language-model training data are

clear examples: when normalizing the spread by mean BPB they have the smallest and third-smallest coefficient of variation in FLORES+ BPB across tokenizers. Chinese is the main exception, despite having the sixth-largest share, it has the largest spread and the largest exclusion penalty. We find that this ranking is sensitive to measurement strategy, and under the alternative approach in §C, Bengali and Tamil, the two lowest resource languages, have a larger spread than Chinese.

Excluding a language from the tokenizer-training data raises its FLORES+ BPB for all 31 languages; the penalty is negatively correlated with the language's share of the language-model training data. We train four BPE tokenizers, denoted *coverage tokenizers*, that differ only in which of the 31 languages are included in their tokenizer-training data.. Their language coverage is nested expanding coverage by resource tiers: English only; the six, 21, or all 31 languages with the largest shares of the language-model training data. For each language, we average its FLORES+ BPB separately over the coverage tokenizers whose tokenizer-training data exclude and include that language. We define the language's *exclusion penalty* as the excluded mean minus the included mean[1]. The exclusion penalty is positive for all 31 languages[2], with Chinese (0.292) and Korean (0.089) having the largest penalties. Fig. 1 (right) plots exclusion penalty against log language-model training-data share; the correlation is negative (Spearman $\rho = -0.33$), but not significant. The relationship becomes significant, though, under either of two controls: restricting the analysis to the 15 languages excluded by at least two coverage tokenizers ($\rho = -0.63$), and controlling for a language's average per-byte information content ($\rho = -0.36$), which varies greatly due to differences in the average number of UTF-8 bytes per character across different scripts (§C). We perform further analyses on potential confounds for this result in §C.

The languages most sensitive to tokenizer choice largely share their tokenizer ranking, so a single tokenizer can be close to the best choice for all of them at once. §3 shows that no single tokenizer ranks first for every language. For the high-resource languages a lower rank corresponds to a small difference in BPB, because their BPB spread across the tokenizers is the smallest. For the 10 languages with the smallest shares of the language-model training data, where the spread is largest, the per-language rankings largely agree: the FLORES+ BPB values of every pair of these 10 languages are positively correlated across the 54 tokenizers (45 pairs, mean Spearman $\rho = 0.56$), and the tokenizer that ranks first on the 31-language FLORES+ mean ranks no lower than 11th of 54 on each of the 10 (BPE Clean plus3 NFC on the balanced data, Table 1). Among the coverage tokenizers above, going from the tokenizer trained on the six high-resource languages to the tokenizer trained on all 31 (Table 5) lowers FLORES+ BPB for 30 of the 31 languages, most for Chinese (0.082), and changes English BPB and validation BPB by less than their seed variation; the exception is Japanese, whose BPB rises by 0.002.

5 Do design choices change per-language outcomes? (RQ3)

Finding 3. Training the tokenizer on only English raises FLORES+ BPB for 30 of the 31 languages, having more of an impact than any other single configuration change we tested. We test two reassignments of the tokenizer training data sampling weights that raise lower-resource languages' shares of the tokenizer training data. Under both, the mean FLORES+ BPB of these languages is *higher*, not lower.

Restricting the tokenizer-training data to English changes BPB more than any other single configuration change. We compare 25 pairs of models where a single design choice in the tokenizer is changed. Table 5 (§D) reports the pairs' difference in performance on the 31-language FLORES+ mean and validation BPB. The largest of the 25 differences comes from restricting the tokenizer-training data to English (+0.0386 in the 31-language FLORES+ mean). Restricting the tokenizer-training data to just the high-resource languages[3], or leaving out only the ten languages with the smallest shares of the language-model training data both result in smaller performance degradations. Of the 25 design choices ablated, 18 show a difference larger than run-to-run variation on the 31-language FLORES+ mean (16 on validation BPB). We discuss the others in §D.

[1] No coverage tokenizer excludes English, so for English we compare the variant of §5 without English web text in its tokenizer-training data with the four coverage tokenizers.

[2] This partly reflects a general negative effect of narrower tokenizer training data coverage: tokenizers that exclude more languages have higher mean FLORES+ BPB across all 31 languages (Table 7 in §E).

[3] We define the high-resource language set as the six languages with the largest shares of the language-model training data (English, Russian, Spanish, French, German, and Chinese).

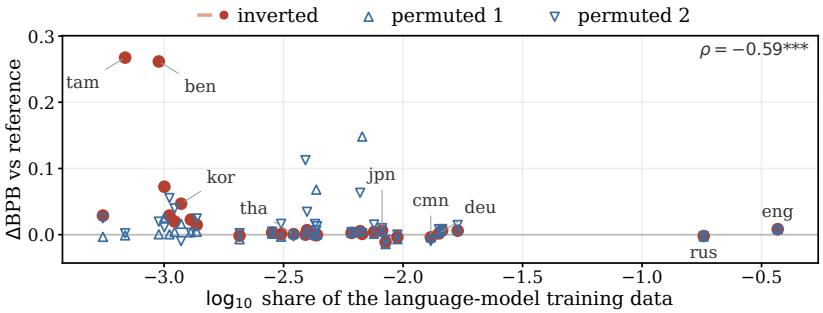

Figure 2: Per-language change in FLORES+ BPB relative to reference (the model trained with the GPT-4o BPE reference tokenizer) for the 31 languages of the training mixture, as a function of the language's log share of the language-model training data (realized byte share). Filled circles: inverted-allocation tokenizer, in which languages with the smallest nominal sampling weights in the language-model training data receive the largest shares of the tokenizer-training data. Open triangles: the two permuted allocation tokenizers, in which tokenizer training data share amounts are assigned randomly.

Raising the tokenizer-training weight of the low-resource languages to an equal share raises their BPB instead of lowering it. For five tokenizers in our main set, we train a version where all 30 FineWeb2 languages are sampled at the same weight instead of weights proportional to each language's amount of FineWeb2 data; English, drawn from FineWeb-Edu, keeps its baseline sampling weight (0.35). The exact protocol is described in §E. We compare pairwise performance; values are given in Table 6.[4]. In all five pairs, the average FLORES+ BPB over the 15 languages with the smallest proportional tokenizer-training weights is higher under the equal weighting scheme. Further, the 31-language FLORES+ mean is also higher. For four of the tokenizers, this rise is larger than run-to-run variation.

Every change to the languages' shares of the tokenizer-training data raised the 31-language FLORES+ mean, and when the languages with the least language-model training data received the largest shares, FLORES+ BPB rose most for the languages with smaller shares of language-model training data. Under the equal weights the shares of all languages change at once, so we cannot observe what happens to a single language whose share of the tokenizer-training data rises. We therefore train four additional tokenizers ablating just the languages' shares of the tokenizer-training data. We use the balanced GPT-4o BPE tokenizer as our reference point, referring to it as the *reference tokenizer* and to its corresponding language model as the *reference model*. In what we call the *inverted-allocation tokenizer,* the languages with the smallest nominal sampling weights in the language-model training data receive the largest shares of the tokenizer-training data; Tamil receives the largest. The other three configurations are one without English web text and two with random permutations of the same shares computed for the inverted-allocation tokenizer (permuted allocations 1 and 2). We train the models of the four additional tokenizers for more steps than the reference model, so that every model is trained on approximately the same total amount of underlying text. Results are similar when we instead keep the number of training tokens matched (§E). Under each of the four targeted configurations, the 31-language FLORES+ mean is higher than the reference model's by more than run-to-run variation (largest rise: $+0.0256$, inverted-allocation tokenizer). Under the inverted-allocation tokenizer, the change in FLORES+ BPB is larger for languages with a smaller share of the language-model training data (Spearman correlation -0.59; Fig. 2). Tamil, which receives the largest share of the tokenizer-training data, has the largest rise in FLORES+ BPB of the 31 languages, $+0.267$. The two permuted allocations give us a reference point: the same correlation is $+0.08$ and -0.28, respectively, neither significant (Table 8 in §E). Together with the equal-weighting results, this suggests that allocating vocabulary entries to a language is not always helpful to that language (and is perhaps actually harmful). We suspect this may be a by-product of insufficient language-model training data in the respective languages to fit these added entries' embeddings.

[4]For any given pair, both language models are trained on the same number of tokens, but because different tokenizers will segment the underlying text differently, the training bytes across the models differs.

When language-model training data is not repeated, Tamil's FLORES+ BPB rises about half as much under the tokenizer with inverted data allocation, and FLORES+ BPB no longer rises most for the languages with smaller shares of language-model training data. Every model discussed so far makes two to three passes over the language-model training mixture. This is arguably the more common setting for language model training on lower-resource languages, where data must be repeated Muennighoff et al. (2023). To test the impact of this design choice, we train models for the reference tokenizer and the inverted-allocation tokenizer on a larger language-model training mixture of the same composition, in which no document is processed twice, i.e., single-pass training. In this regime, Tamil's FLORES+ BPB under a model trained with the inverted-allocation tokenizer is still significantly higher than a model trained with the reference tokenizer ($+0.14$ vs. $+0.267$ with repeated data). However, in single-pass training, the correlation between a language's change in FLORES+ BPB and its share of the language-model training data is positive ($+0.34$) and not significant. The association between a language's share of the language-model training data and its rise in FLORES+ BPB under the inverted-allocation tokenizer therefore appears only when language-model training repeats data. One mechanism that we hypothesize, which our experiments do not isolate, is overfitting of the token embeddings: the inverted-allocation tokenizers segment the low-resource languages into fewer, longer, more language-specific vocabulary entries (Tamil's fertility falls to 0.379 times the reference tokenizer's; §E). These tokens necessarily occur in a smaller set of training contexts, and repeating those contexts may overfit the embeddings to them. A check in §E on per-vocabulary-entry losses limits this explanation: the rise in loss is spread across vocabulary entries rather than concentrated on rarely trained ones.

6 Which tokenizer properties matter for which language? (RQ4)

Finding 4. No intrinsic tokenizer metric trends with BPB the same way for every language. Within a language, there are consistent and significant trends; which of these relationships holds, and how strongly, changes with the language's share of the language-model training data. Using intrinsic metrics measured on a per-language basis, we can train pairwise tokenizer BPB rankers that reach an accuracy of 0.89, suggesting a viable route for screening tokenizers during development.

Disclaimer: Every relationship between a metric and FLORES+ BPB or validation BPB in this section is a correlation across tokenizers, and not a causal claim.

Within a language, all tokenizer intrinsic metrics except vocabulary utilization are significantly related to FLORES+ BPB and MultiBLiMP accuracy. We fit the linear mixed-effects models described in §2.3, which have one intercept per language, once for each of the nine intrinsic tokenizer metrics (defined in §F.1). Both the outcome variable and the intrinsic metric are computed on a per-language basis. With per-language FLORES+ BPB as the outcome and per-language intrinsic metrics measured on FLORES+, eight of the nine coefficients corresponding to the different intrinsic metrics are significant (Table 11 in §F.1). Tokenizers with a higher unigram, bigram, or trigram entropy, a higher Rényi efficiency, or a higher compression rate on a language have a lower FLORES+ BPB on that language. Tokenizers with a higher fertility, a higher character-split rate, or a higher boundary-crossing rate have a higher FLORES+ BPB. The coefficient of vocabulary utilization is not significant. With per-language MultiBLiMP accuracy as the outcome, on the 24 languages that MultiBLiMP covers, the same eight coefficients are significant, each with the reversed sign (since higher is better for this metric). With intrinsic metrics and BPB computed on FineWeb2 text, all nine coefficients are significant, and the coefficient of vocabulary utilization is *negative* (§F.2). Notably, when we adjust each run's FLORES+ BPB for its number of training bytes in §F.3, the coefficient of compression rate is significant under only one of our four estimates of the adjustment. This suggests that several previous results in the literature finding that tokenizer compression rate correlates with model performance (§1) were perhaps confounded by differences in underlying training bytes across the models tested.

For single languages, different metrics are significantly correlated with FLORES+ BPB. For each of the 31 languages and each intrinsic metric, we compute the Spearman correlation between the metric and FLORES+ BPB across the 54 tokenizers. Which metrics correlate significantly with downstream performance differs greatly across languages. Correlation heatmaps visualize this result in §F. As concrete examples, the correlation of Rényi efficiency is significant for Greek (-0.72) and Bengali (-0.54), and near zero for Korean (-0.02) and Tamil ($+0.17$). As another example, three of the four metrics that correlate significantly with BPB for Tamil (compression rate, unigram entropy, and fertility) have correlation coefficients of the opposite sign from the mixed-effect model

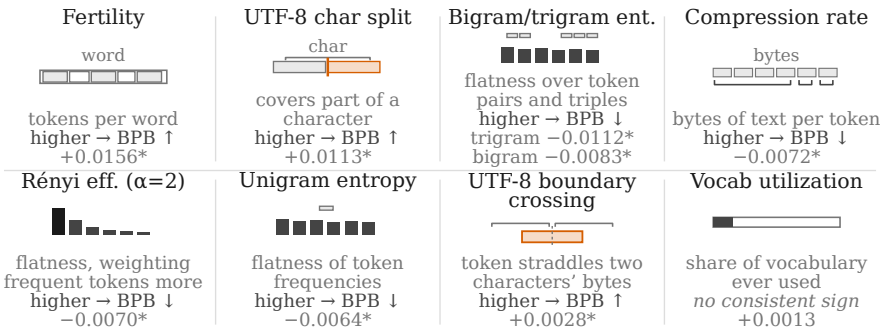

Figure 3: The intrinsic metrics we consider and their within-language relationships with FLORES+ BPB (lower ↓ better). Each tile gives the metric's definition, the direction of the relationship, and the coefficient from the within-language model of §2, in BPB per standard deviation of the metric.

coefficients (Table 11), which quantify the average within-language relationship between the metric and per-language BPB. This shows that single languages can benefit from tokenizer properties quite different from what helps on average. In §F.2, we see the results are robust to choice of evaluation corpus, and in §F.3, we only find a significant interaction between the metric and the language's share of the language-model training data for two of the nine metrics,suggesting these relationships are largely properties of the language rather than artifacts of the model training setup. Collectively, we take this as evidence that different tokenizer properties are important for different languages.

Measuring intrinsic metrics per language, rather than as one value per tokenizer, enables tokenizer ranking without model training: 0.89 **pairwise accuracy on held-out tokenizer families.** We ask whether intrinsic tokenizer metrics can be used to rank tokenizers by their eventual BPB, allowing us to skip language model training for tokenizer assessment. We train the logistic models described in §2.3, using the intrinsic metrics as features. We score their pairwise ranking accuracy: the fraction of held-out tokenizer pairs that a metric-based ranker orders the same way as the trained models' BPB. In each cross-validation fold, we hold out one of the 42 tokenizer families, so a held-out tokenizer is never ranked by a model fit on members of its own family. With the nine metrics as one value per tokenizer, these classifiers order held-out tokenizer pairs according to their validation BPB at accuracy 0.700 and according to their 31-language FLORES+ mean BPB at accuracy 0.735. With the same nine metrics measured separately on each of the 31 languages, the accuracies rise to 0.907 and 0.890, respectively (Table 16). These accuracies are higher than those from ranking the tokenizers by the validation BPB of smaller models trained with them (Table 16). Ranking by intrinsic metrics can also be done accurately on a per-language basis: when predicting pairwise ranks of individual language's own FLORES+ BPB, the per-language ranker scores between 0.738 (Czech) and 0.878 (Tamil). These results indicate that intrinsic metrics can be used for screening tokenizer choice, both at the aggregate and per-language level.

7 Recommendations

Three recommendations follow from the answers to the four questions (§ 3 to 6).

- *Per-language evaluation.* Evaluate candidate tokenizers on each language the model is meant to serve, or at least stratified by resource tiers and writing systems. Tokenizer rankings agree only partially across languages (§3), and aggregate or English-only evaluations give little or no weight to the languages with the least language-model training data, which are the languages most sensitive to the tokenizer (§4).
- *Caution with reallocating tokenizer capacity toward low-resource languages.* Every reallocation we tested raised the BPB of the lower-resource languages that it was meant to favor, and every departure from proportional training data weights raised the 31-language FLORES+ mean BPB (§5). We therefore recommend proportional tokenizer-training weights as the default; targeted allocations of the vocabulary to lower-resource languages is an intervention that should be validated, not presumed beneficial.
- *Metric-based screening before training.* Use intrinsic metrics, measured per language, to screen candidate tokenizers before any model training as they provide good signal for tokenizer quality (§6). Importantly, these metrics should be used as screening statistics, not training objectives.

Reproducibility Statement

§B.1 lists every tokenizer with its exact configuration, vocabulary size, and family assignment, and §B.2 describes each pretokenization regex. §B.3 states the full architecture, optimizer settings, and step counts, and §B.4 the composition of the language-model training mixture with per-language byte shares (Table 4). §B.5 specifies the BPB computation, the seed-variation estimates, and the clustered standard-error construction, and §F.4 the ranking protocol, which is deterministic and uses no random seeds. All evaluation data are public (FLORES+, MultiBLiMP, XCOPA, Belebele). Every number in the paper is bound, in the paper source, to a named key of a result artifact computed programmatically from the run outputs. We will release the tokenizers, model checkpoints, result artifacts, and the training, evaluation, and analysis code upon de-anonymization.

The Use of Large Language Models

Large language model (LLM) based tools were used in two ways in this work. First, for writing assistance: polishing author-written text for grammar and clarity and checking terminology for consistency. Second, as agents assisting with experiment management, for example launching training runs, monitoring them, and collecting their outputs into result files. The research questions, experimental design, analyses, and conclusions are the authors' own. Every number reported in this paper is bound, in the paper source, to a named key of a result artifact that was computed programmatically from the run outputs, and the authors checked these bindings against the underlying artifacts to ensure that no LLM-introduced errors (hallucinated or mistranscribed values) are present in the reported results.

References

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 Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9904–9923, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.614. URL https://aclanthology.org/2023.emnlp-main.614/.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. Tokenizer choice for LLM training: Negligible or crucial? In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3907–3924, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.247. URL https://aclanthology.org/2024.findings-naacl.247/.

Gül Sena Altıntaş, Malikeh Ehghaghi, Brian Lester, Fengyuan Liu, Wanru Zhao, Marco Ciccone, and Colin Raffel. TokSuite: Measuring the impact of tokenizer choice on language model behavior. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2512.20757.

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

Catherine Arnett, Tyler A. Chang, Stella Biderman, and Ben Bergen. Explaining and mitigating crosslingual tokenizer inequities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

Ehsaneddin Asgari, Yassine El Kheir, and Mohammad Ali Sadraei Javaheri. Morphbpe: A morpho-aware tokenizer bridging linguistic complexity for efficient llm training across morphologies. *CoRR*, abs/2502.00894, February 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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 749–775, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.44. URL https://aclanthology.org/2024.acl-long.44/.

Gautier Dagan, Gabriel Synnaeve, and Baptiste Rozière. Getting the most out of your tokenizer for pre-training and domain adaptation, February 2024. URL http://arxiv.org/abs/2402.01035. arXiv:2402.01035 [cs].

Negar Foroutan, Clara Meister, Debjit Paul, Joel Niklaus, Sina Ahmadi, Antoine Bosselut, and Rico Sennrich. Parity-aware byte-pair encoding: Improving cross-lingual fairness in tokenization. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7514–7538, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.342. URL https://aclanthology.org/2026.acl-long.342/.

Matthias Gallé. Investigating the effectiveness of BPE: The power of shorter sequences. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1375–1381, Hong Kong, China, November 2019. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL https://arxiv.org/abs/2101.00027.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2274–2286, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

Alejandro Hernández-Cano, Alexander Hägele, Allen Hao Huang, Angelika Romanou, Antoni-Joan Solergibert, Barna Pásztor, Bettina Messmer, Dhia Garbaya, Eduard Frank Ďurech, Ido Hakimi, Juan Garcia Giraldo, Mete Ismayilzada, Negar Foroutan, Skander Moalla, Tiancheng Chen, Vinko Sabolčec, Yixuan Xu, Michael Aerni, Badr AlKhamissi, Inés Altemir Marinas, Mohammad Hossein Amani, Matin Ansaripour, Ilia Badanin, Harold Benoit, Emanuela Boros, Nicholas John Browning, Fabian Bösch, Maximilian Böther, Niklas Canova, Camille Challier, Clément Charmillot, Jonathan Coles, Jan Milan Deriu, Arnout Devos, Lukas Drescher, Daniil Dzenhaliou, Maud Ehrmann, Dongyang Fan, Simin Fan, Silin Gao, Miguel Gila, María Grandury, Diba Hashemi, Alexander Miserlis Hoyle, Jiaming Jiang, Mark Klein, Andrei Kucharavy, Anastasiia Kucherenko, Frederike Lübeck, Roman Machacek, Theofilos Ioannis Manitaras, Andreas Marfurt, Kyle Matoba, Simon Matrenok, Henrique Mendonça, Fawzi Roberto Mohamed, Syrielle Montariol, Luca Mouchel, Sven Najem-Meyer, Jingwei Ni, Gennaro Oliva, Matteo Pagliardini, Elia Palme, Andrei Panferov, Léo Paoletti, Marco Passerini, Ivan Pavlov, Auguste Poiroux, Kaustubh Ponkshe, Nathan Ranchin, Javier Rando, Mathieu Sauser, Jakhongir Saydaliev, Mukhammadali Sayfiddinov, Marian Schneider, Stefano Schuppli, Marco Scialanga, Andrei Semenov, Kumar Shridhar, Raghav Singhal, Anna Sotnikova, Alexander Sternfeld, Ayush Kumar Tarun, Paul Teiletche, Jannis Vamvas, Xiaozhe Yao, Hao Zhao, Alexander Ilic, Ana Klimovic, Andreas Krause, Caglar Gulcehre, David Rosenthal, Elliott Ash, Florian Tramèr, Joost VandeVondele, Livio Veraldi, Martin Rajman, Thomas C. Schulthess, Torsten Hoefler, Antoine Bosselut, Martin Jaggi, and Imanol Schlag. Apertus: Democratizing open and compliant LLMs for global language environments. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 46877–46955, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2172. URL https://aclanthology.org/2026.acl-long.2172/.

Haris Jabbar. Morphpiece : A linguistic tokenizer for large language models, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. GitHub repository/blog post.

Jaap Jumelet, Leonie Weissweiler, Joakim Nivre, and Arianna Bisazza. MultiBLiMP 1.0: A massively multilingual benchmark of linguistic minimal pairs. *Transactions of the Association for Computational Linguistics (TACL)*, 2026.

Andrej Karpathy. nanochat, 2025. URL https://github.com/karpathy/nanochat. Open-source repository and model release.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL https://aclanthology.org/P18-1007/.

Sander Land. Mingram: A minimalist unigram tokenizer with high compression and competitive morphological alignment, 2026. URL https://arxiv.org/abs/2606.27019.

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

Pietro Lesci, Clara Meister, Thomas Hofmann, Andreas Vlachos, and Tiago Pimentel. Causal estimation of tokenisation bias. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28325–28340, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1374. URL https://aclanthology.org/2025.acl-long.1374/.

Brian Lester, Jaehoon Lee, Alexander A Alemi, Jeffrey Pennington, Adam Roberts, Jascha Sohl-Dickstein, and Noah Constant. Training LLMs over neurally compressed text. *Transactions on Machine Learning Research*, 2024.

Raymond Li, Loubna Ben allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia LI, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Joel Lamy-Poirier, Joao Monteiro, Nicolas Gontier, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Ben Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason T Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Urvashi Bhattacharyya, Wenhao Yu, Sasha Luccioni, Paulo Villegas, Fedor Zhdanov, Tony Lee, Nadav Timor, Jennifer Ding, Claire S Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro Von Werra, and Harm de Vries. Starcoder: may the source be with you! *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=KoFOg41haE. Reproducibility Certification.

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 (COLM 2025)*, 2025. URL https://arxiv.org/abs/2503.13423.

Jonas F. Lotz, António V. Lopes, Stephan Peitz, Hendra Setiawan, and Leonardo Emili. Beyond text compression: Evaluating tokenizers across scales. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32155–32173, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1546. URL https://aclanthology.org/2025.acl-long.1546/.

Anton Lozhkov, Loubna Ben Allal, Elie Bakouch, Leandro von Werra, and Thomas Wolf. FineMath: the finest collection of mathematical content. 2024. URL https://huggingface.co/datasets/HuggingFaceTB/finemath.

Clara Meister. Tokeval: A tokenizer analysis suite. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=RpcJS02Wcb.

Niklas Muennighoff, Alexander M Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=j5BuTrEj35.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No language left behind: Scaling human-centered machine translation. 2022.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. Fineweb2: One pipeline to scale them all — adapting pre-training data processing to every language. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=jnRBe6zatP.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 36963–36990. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/74bb24dca8334adce292883b4b651eda-Paper-Conference.pdf.

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

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

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*, pp. 678–702, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.40. URL https://aclanthology.org/2024.emnlp-main.40/.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162/.

Aaditya K. Singh and DJ Strouse. Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs. *arXiv preprint arXiv:2402.14903*, 2024.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17084–17097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8df7c2e3c3c3be098ef7b382bd2c37ba-Paper.pdf.

Xiulin Yang, Ethan Gotlieb Wilcox, and Catherine Arnett. Apples to apples? towards comparable crosslingual language model evaluation, 2026. URL https://arxiv.org/abs/2608.25089.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5184–5207, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.284. URL https://aclanthology.org/2023.acl-long.284/.

A LIMITATIONS

Our results use one architecture (nanochat d24, decoder-only) at 1.27B parameters, one training recipe, one vocabulary size near 128K, and, for most of the 54-tokenizer set, one run per tokenizer. Our primary outcome is language-modeling performance (BPB). Of the task benchmarks we evaluated, only MultiBLiMP separates the models: XCOPA and Belebele accuracies stay near chance at this scale (§G), so our conclusions concern language-modeling performance, not downstream task ability. The 54 tokenizers are an observational panel rather than a randomized design: the metric analyses of §6 are correlations across this panel, and a metric that co-varies with the ablated design choices cannot be fully separated from them. Every run in our main analysis makes two to three passes over the language-model training mixture. Aggregate model rankings change little without repetition or at twice the token budget (Spearman $+0.93$ and $+0.94$ over 11 runs; §C), but several per-language results interact with repetition, e.g., the reallocation experiments (§5). Finally, the 31 languages are those of our training mixture; statements about which languages are most sensitive to tokenizer choice are statements within this set.

B EXPERIMENTAL DETAILS

B.1 TOKENIZER CONFIGURATIONS

This section expands on §2, where Table 1 lists the 54 tokenizers with their configuration and their family (§2); a tokenizer without a family code forms a single-member family.

Tokenizer families. We define tokenizer families as groups of tokenizers differing in only minor hyperparameter settings. Differences in algorithm, pretokenization regex, normalization, encoding, or training data are treated as larger design changes, and so we categorize the tokenizers differing on these axes into different families. There are two cases that can arguably be seen as exceptions: we group variants of one pretokenization regex that differ in single clauses into one family, and we also group each pair of tokenizers whose training data differ only in sampling settings into one family.

Tokenizer training. We train tokenizers with a vocabulary near 128K on 10 GB of text drawn from the data mixtures used for language model training (§B.4) or their restrictions and reweightings. Most tokenizers are trained using the HuggingFace `tokenizers` library. The SuperBPE, parity-aware BPE, and SCRIPT tokenizers were built with separate training code, based off the codebases released by the respective authors. The off-the-shelf Mistral-Nemo vocabulary is $131{,}072$ (about 1K of them special tokens) and the LLaMA-3 vocabulary is $128{,}256$.

Defective tokenizers. Two implementation defects described in §2.1 affect some panel members, these are marked with footnotes in Table 1. The byte-alphabet defect (marker ^{b}, described in §2) occurs when *the base alphabet of a byte-level tokenizer lacks some of the 256 byte symbols, so those bytes cannot be represented by the initial vocabulary*. Where a defect-free counterpart exists, we use the full 256-entry byte-alphabet build (marker ^{a}). For example, the english-data GPT-4o row is such a re-build, which replaced the original defective tokenizer among the 54. The lone-space drop (marker ^{c}) is that *the decoder's Metaspace convention removes a token consisting only of a space, so a lone space is lost in the decoded text*. We retain the three marked defective tokenizers (b and c) because no defect-free counterpart exists.

Variations of tokenizer-training data coverage. Some of the 54 tokenizers differ only in which languages their training data covers, and one further experiment varies how that data is allocated across languages; we describe both here. The coverage sequence used in §4 and Table 5 is the set of GPT-4o-regex BPE tokenizers with data marked english, highres, highmid, and balanced in Table 1.

For the reallocation experiment (§E), we trained four additional tokenizers whose training data is allocated across languages in deliberately unusual ways. In the inverted-allocation tokenizer, the languages with the least language-model training data receive the largest shares of the tokenizer-training data; in the two permuted allocations, the same 31 shares are assigned in two random orders to the original balanced mixture. We exclude these four tokenizers from the 54 tokenizers in the main analyses, as such allocations are unrealistic for tokenizers meant for practical use, and including them would skew the statistics that we compute over the main panel, such as the standard deviation of each language's BPB across tokenizers and the ranking of tokenizers within each language.

Table 1: The 54 tokenizers. Pretok: pretokenization regex (§B.2). Data: tokenizer-training mixture (balanced multilingual; english-only; code-heavy; the highres / highmid coverage subsets; all-equal reweighting; balanced-half; multi-heavy; the full-language-pool FineWeb2 sample, fw2full; off-the-shelf tokenizers are not trained by us). Norm.: Unicode normalizer. Family: tokenizers sharing a code form one family under the rule of §2; codes abbreviate the identifiers of the family partition, and a blank entry is a single-member family. Vocab: the tokenizer's vocabulary size including special tokens, read through the evaluation loader; this is not the padded model embedding size. [a]full 256-entry byte-alphabet rebuild (described in §D); [b]byte-alphabet defect (described in §D); [c]lone-space drop through the decoder's Metaspace convention (§2); [s]SCRIPT text encoding (§2); [d]NFD, followed by the deletion of the control characters CR, VT, FF, and NEL and the replacement of Unicode space characters by the ASCII space (`U+0020`).

Tokenizer	Algorithm	Pretok	Data	Norm.	Vocab	Family
BPE GPT-4o (balanced)	BPE	GPT-4o	balanced	—	128,256	G
BPE GPT-4o (balanced, file cap)	BPE	GPT-4o	balanced	—	128,256	G
BPE GPT-4o NFC (balanced)	BPE	GPT-4o	balanced	NFC	128,256	
BPE GPT-4o (english)[a]	BPE	GPT-4o	english	—	128,256	
BPE GPT-4o (code)	BPE	GPT-4o	code	—	128,256	
BPE GPT-4o (highres)	BPE	GPT-4o	highres	—	128,256	
BPE GPT-4o (highmid)	BPE	GPT-4o	highmid	—	128,256	
BPE GPT-4o (all-equal)	BPE	GPT-4o	all-equal	—	128,256	fam05
BPE GPT-4o (all-equal, no repeat)	BPE	GPT-4o	all-equal	—	128,256	fam05
BPE GPT-4o NFC (all-equal)	BPE	GPT-4o	all-equal	NFC	128,256	
BPE GPT-4o NFC (fw2full)	BPE	GPT-4o	fw2full	NFC	128,000	
BPE GPT-4o script-enc. (bal.)[s]	BPE	GPT-4o	balanced	NFC	128,256	
BPE Claude (balanced)	BPE	Claude	balanced	—	128,256	
BPE Claude NFC (balanced)	BPE	Claude	balanced	NFC	128,256	
BPE Claude (english)[b]	BPE	Claude	english	—	128,256	
BPE Claude (all-equal)	BPE	Claude	all-equal	—	128,256	
BPE Claude V2 (balanced)	BPE	Claude V2	balanced	NFD[d]	128,256	
BPE GPT-2 (balanced)	BPE	GPT-2	balanced	—	128,256	
BPE Punct (balanced)	BPE	Punct	balanced	—	128,256	
BPE Punct (english)[b]	BPE	Punct	english	—	128,256	
BPE Punct (all-equal)	BPE	Punct	all-equal	—	128,256	
BPE RightAlign (balanced)	BPE	RightAlign	balanced	—	128,256	
BPE RightAlign NFC (balanced)	BPE	RightAlign	balanced	NFC	128,256	
BPE Whitespace (balanced)	BPE	Whitespace	balanced	—	128,256	
BPE Whitespace (multi-heavy)	BPE	Whitespace	multi-heavy	—	128,256	
BPE Clean NFC (balanced)	BPE	Clean	balanced	NFC	128,004	fam00
BPE Clean no-mark NFC (bal.)	BPE	Clean no-mark	balanced	NFC	128,000	fam00
BPE Clean plus2 NFC (bal.)	BPE	Clean plus2	balanced	NFC	128,256	fam00
BPE Clean plus3 NFC (bal.)	BPE	Clean plus3	balanced	NFC	128,256	fam00
BPE Clean NFC (fw2full)	BPE	Clean	fw2full	NFC	128,000	
Mistral-Nemo	BPE	Mistral-Nemo	off-the-shelf	—	131,072	
LLaMA-3	BPE	LLaMA-3	off-the-shelf	—	128,256	
Unigram GPT-4o (balanced)	UnigramLM	GPT-4o	balanced	—	128,256	D
Unigram GPT-4o tuned (balanced)	UnigramLM	GPT-4o	balanced	—	128,256	D
Unigram GPT-4o (highres)	UnigramLM	GPT-4o	highres	—	128,256	
Unigram GPT-4o (highmid)	UnigramLM	GPT-4o	highmid	—	128,256	
Unigram Claude (balanced)	UnigramLM	Claude	balanced	—	128,256	
Unigram RightAlign (balanced)	UnigramLM	RightAlign	balanced	—	128,256	
Unigram Whitespace (balanced)[c]	UnigramLM	Whitespace	balanced	—	128,256	
MinGram script-enc. (balanced)[s]	MinGram	SCRIPT	balanced	NFC	128,256	fam06
MinGram script-enc., split newlines[s]	MinGram	SCRIPT	balanced	NFC	128,256	fam06
SuperBPE GPT-4o (balanced)	SuperBPE	GPT-4o	balanced	—	128,004	
SuperBPE Clean C2 (balanced)	SuperBPE	Clean	balanced	NFC	128,004	C
SuperBPE Clean C3 (balanced)	SuperBPE	Clean	balanced	NFC	128,004	C
SuperBPE Punct t64k (bal.-half)	SuperBPE	Punct	balanced-half	—	128,004	
SuperBPE/PA-BPE GPT-4o (balanced)	SuperBPE/PA-BPE	GPT-4o	balanced	NFC	128,004	B
SuperBPE/PA-BPE GPT-4o t64k	SuperBPE/PA-BPE	GPT-4o	balanced	NFC	128,004	B
PA-BPE GPT-4o (balanced)	PA-BPE	GPT-4o	balanced	NFC	127,826	A
PA-BPE GPT-4o hybrid-window (bal.)	PA-BPE	GPT-4o	balanced	NFC	127,826	A
PA-BPE Clean (balanced)	PA-BPE	Clean	balanced	NFC	127,836	
PA-BPE GPT-4o (fw2full)	PA-BPE	GPT-4o	fw2full	NFC	127,825	E
PA-BPE GPT-4o hybrid-win. (fw2full)	PA-BPE	GPT-4o	fw2full	NFC	127,825	E
PA-BPE Clean (fw2full)	PA-BPE	Clean	fw2full	NFC	127,835	F
PA-BPE Clean hybrid-win. (fw2full)	PA-BPE	Clean	fw2full	NFC	127,835	F

B.2 PRETOKENIZATION REGEXES

A pretokenizer splits raw text into chunks before the subword algorithm merges characters within chunks; it never merges across chunk boundaries, so it sets an upper bound on which tokens can form. The 54 tokenizers use the following pretokenizers. All of them are byte-level except the Whitespace UnigramLM tokenizer and the three SCRIPT tokenizers. The Whitespace UnigramLM tokenizer uses the Metaspace pretokenizer with byte fallback, and the SCRIPT tokenizers use the script-class encoding.

- **GPT-4o** (the `o200k` regex): splits text into CamelCase-aware runs of letters with optional English contraction suffixes (`'s`, `'t`, and so on), runs of one to three digits, runs of punctuation (optionally preceded by a single space), and whitespace. It is Unicode-property aware (`\p{L}`, `\p{N}`) and is the default for recent OpenAI tokenizers.

- **GPT-2**: the original GPT-2 regex applied by the byte-level pretokenizer. Similar in spirit to GPT-4o but older and ASCII-oriented; it does not cap digit runs at three, so long numbers are grouped differently.

- **Claude**: reverse-engineered from the Claude tokenizer. A close variant of GPT-4o that differs in several ways, among them the following three: a word takes only a leading space as prefix (not any leading non-letter, non-digit character); punctuation runs do not absorb trailing newlines; and whitespace is split by type (spaces, tabs, newlines, and carriage returns each grouped separately) rather than as one whitespace class. It shares GPT-4o's CamelCase grouping and its one-to-three-digit groups. Unlike GPT-4o, it has a separate branch that matches a contraction suffix on its own, and it accepts the curly apostrophe in contraction suffixes.

- **Claude V2**: a revision of the Claude pattern from a re-probe of the production tokenizer, used by one balanced BPE tokenizer together with an NFD normalizer (Table 1). It drops the CamelCase clauses (a word is a single run of letters and combining marks), makes the contraction suffix its own clause, and attaches a leading space to the following letter or punctuation run only when the preceding character is not a digit, whitespace, Han, or Hangul.

- **Clean**: a simplified multilingual variant of the GPT-4o pattern with a space-only word prefix that splits each digit individually rather than in groups.

- **Clean no-mark**: the Clean pattern with combining marks (`\p{M}`) removed from the letter classes, so a combining mark no longer stays with its base letter and splits off into the punctuation branch. Its tokenizer differs only in this pretokenizer from a Clean BPE tokenizer that we built alongside it and that is not one of the 54 tokenizers. It also differs from the balanced Clean BPE tokenizer of Table 1, which we built earlier, in its vocabulary size (128,000 vs. 128,004).

- **Clean plus2 / plus3**: the Clean pattern with an attachment class on the word branches: a leading space, Tibetan tsheg, or ASCII or curly apostrophe attaches to the following word. Plus3 additionally attaches a word-final apostrophe not followed by a letter. Both cap punctuation and whitespace runs at 16 characters.

- **RightAlign**: GPT-4o modified so that multi-digit numbers split into three-digit groups aligned from the right ("123456" becomes "123", "456" rather than a left-aligned grouping), intended to align place value for arithmetic.

- **Punctuation**: isolates every punctuation character as its own chunk before the byte-level regex, so punctuation never merges with adjacent text.

- **Whitespace**: the minimal pretokenizer. It splits only on whitespace, with no further regex, so a chunk is a whole whitespace-delimited word; case, digit, and punctuation boundaries inside a word are left to the subword algorithm. The two Whitespace BPE tokenizers use this

pattern. The Whitespace UnigramLM tokenizer instead uses the Metaspace pretokenizer, which splits only on spaces and not on tabs or line breaks.

B.3 Model Architecture and Training Details

Architecture. All models use the nanochat d24 architecture, a decoder-only transformer with 24 layers totaling 1.27B parameters total at our 128K vocabulary size. Each model is trained on 4 NVIDIA GH200 GPUs in `bfloat16`, with a context length of 2048, resulting in an effective batch size of $2^{20} = 1{,}048{,}576$ tokens. The architecture in full: hidden dimension 1536, 12 attention heads (head dimension 128), RoPE (Su et al., 2024) with base frequency $\theta = 100{,}000$, squared-ReLU activations, parameterless RMSNorm, QK-norm with scale 1.2, and logit soft-capping at $15 \cdot \tanh(\cdot/15)$; attention uses an "SSSL" sliding-window pattern (three windowed layers of $\lceil L/4 \rceil$ tokens followed by one full-context layer) and value embeddings on alternating layers; the 1.27B parameters divide into 682M transformer matrices and 197M each for input embeddings, output projection, and value embeddings.

Optimization. Weight matrices use Muon (Jordan et al., 2024); embeddings, projections, and scalars use AdamW; learning rates transfer across widths with μP (Yang et al., 2021) and are scaled by $\sqrt{B/B_{\text{ref}}} = \sqrt{2}$ at the batch size $B = 2^{20}$. The learning rate warms up over 40 steps, holds to 35% of training, then decays linearly to 5% of peak; Muon momentum runs $0.85 \rightarrow 0.97$ over 400 steps and back to 0.90 during warm-down; weight decay decays cosine to zero. Per-group effective settings are reported in Table 2.

We set the token budget to nanochat's scaling-law ratio of 10.5 tokens per parameter in the transformer matrices and the output projection. This gives ≈ 9.2B tokens, or about $8{,}800$ steps ($8{,}794$ to $8{,}843$, depending on the vocabulary size). In each run, the data loader makes between about 2.0 and 2.9 passes over the documents of the language-model training mixture; we call one read of the mixture by the data loader a *pass*, and the pass counts stated elsewhere in the paper are these loader passes. The data loader crops a document that does not fit the space that remains in a sequence. The text that a model is trained on therefore amounts to between about 1.1 and 2.0 times the text of the mixture (§E measures the effect of this repetition).

The smaller ranking-proxy models (nanochat d8, d12, and d16: 8, 12, and 16 layers; hidden width 512, 768, 1024; 4, 6, 8 heads; 0.22B, 0.38B, 0.60B total parameters; Table 3) share the d24 data, tokenizers, and optimizer recipe, with μP transferring learning rates across width, and a budget of about 3.5B tokens. The d8 and d12 models are trained for $3{,}337$ steps, and the d16 models for $3{,}337$ to $3{,}370$ steps, depending on the vocabulary size. Their absolute BPB is not comparable to the d24 models (smaller model, smaller token budget); only the induced tokenizer ranking is used, which is invariant to a shift in BPB level.

Table 2: Effective learning rates and optimizer settings per parameter group of the d24 models. *Muon multiplies the learning rate of a matrix by the square root of its ratio of rows to columns when that ratio exceeds one. The learning rate is therefore 0.0566 for the MLP input projection and 0.0980 for the matrices that project the value embeddings to the attention values. The smear parameters mix the embedding of the previous token into the current position, and the backout parameter subtracts the residual stream of the middle layer before the output layer.

Group	Optimizer	Eff. LR	Betas	WD
Transformer matrices	Muon	0.0283*	mom. schedule	0.050→0
Input embeddings	AdamW	0.2121	(0.8, 0.995)	0.001
Output projection	AdamW	0.0057	(0.8, 0.96)	0.01
Value embeddings	AdamW	0.1061	(0.8, 0.995)	0.01
Residual scalars	AdamW	0.0071	(0.8, 0.95)	0.05
Skip scalars	AdamW	0.707	(0.96, 0.95)	0.0
Smear and backout parameters	AdamW	0.2	(0.8, 0.95)	0.0

B.4 Model Training Data

For language-model training data, we use the same corpora as for tokenizer training, in a fixed composition unless otherwise stated. Matching the two keeps the tokenizer's training distribution aligned with the text it later segments. We follow common pretraining practices, training on English

Table 3: Smaller proxy models used for ranking transfer (Table 16). All share the d24 training data, tokenizers, optimizer recipe, and a budget of about 3.5B tokens, which is 3,337 steps for d8 and d12 and 3,337 to 3,370 steps for d16; μP transfers learning rates across width. Parameters are totals at the 128K vocabulary.

Model	Layers	Width	Heads	Params
nanochat d8	8	512	4	0.22B
nanochat d12	12	768	6	0.38B
nanochat d16	16	1024	8	0.60B
nanochat d24 (main)	24	1536	12	1.27B

and multilingual web text with smaller shares of math and coding. The mixture is 36.9% English (FineWeb-Edu), 33.4% multilingual over 30 languages (FineWeb2), 16.1% math (FineMath), and 13.6% code (StarCoderData), totaling 25 GB of UTF-8 text.

Per-language data is sampled in proportion to its presence in FineWeb2; the resulting realized byte share is the resource variable of the sensitivity analysis in §4. Documents are drawn by source weight, without reference to the tokenizer, so the mixture fixes each language's share of the *drawn* bytes. Realized byte shares of the 25 GB mixture range from English (36.9%) and Russian (18.1%) to Catalan (0.06%), with full break-down in Table 4. The mixture does not fix the training bytes: the BOS-bestfit packer discards a tokenizer-dependent 30.4% to 43.5% of the drawn text (median 35.5% over the main census runs), and the discard redistributes the per-language shares of the training bytes tokenizer-dependently rather than by one overall factor: the ratio of a language's share of the training bytes to its share of the drawn bytes deviates from one by up to +38% (Chinese, under the balanced UnigramLM tokenizer) and −23% (Hungarian, under the SuperBPE tokenizer) across five tokenizers trained on the balanced tokenizer-training data, and by −41% (Tamil) under the English-only tokenizer.

Table 4: Per-language data shares of the multilingual portion of the model training mixture: GB of UTF-8 text, realized % of the 25 GB total, and the nominal document-draw weight (w) the sampler used. We draw each document by first choosing a source with probability proportional to its weight and then reading that source's next document. The per-language weights are proportional to estimated character counts. The realized byte shares therefore differ from the weights where mean document length differs across sources. Codes are those of the training-data space; the FLORES+ evaluation pairs cmn_Hani with cmn_Hans. All values from the built mixture's own metadata.

Language	Code	Script	GB	%	w	Language	Code	Script	GB	%	w
Russian	rus_Cyrl	Cyrillic	4.521	18.1	0.1064	Hungarian	hun_Latn	Latin	0.099	0.4	0.0051
Spanish	spa_Latn	Latin	0.423	1.7	0.0245	Vietnamese	vie_Latn	Latin	0.109	0.4	0.0043
German	deu_Latn	Latin	0.353	1.4	0.0238	Czech	ces_Latn	Latin	0.077	0.3	0.0043
French	fra_Latn	Latin	0.364	1.5	0.0206	Greek	ell_Grek	Greek	0.108	0.4	0.0035
Chinese	cmn_Hani	CJK	0.326	1.3	0.0191	Thai	tha_Thai	Thai	0.087	0.3	0.0023
Japanese	jpn_Jpan	CJK	0.204	0.8	0.0127	Finnish	fin_Latn	Latin	0.034	0.1	0.0023
Italian	ita_Latn	Latin	0.165	0.7	0.0119	Slovak	slk_Latn	Latin	0.028	0.1	0.0020
Portuguese	por_Latn	Latin	0.189	0.8	0.0119	Bulgarian	bul_Cyrl	Cyrillic	0.052	0.2	0.0019
Turkish	tur_Latn	Latin	0.152	0.6	0.0094	Korean	kor_Hang	Hangul	0.029	0.1	0.0016
Indonesian	ind_Latn	Latin	0.236	0.9	0.0091	Croatian	hrv_Latn	Latin	0.026	0.1	0.0017
Polish	pol_Latn	Latin	0.107	0.4	0.0088	Catalan	cat_Latn	Latin	0.014	0.1	0.0010
Ukrainian	ukr_Cyrl	Cyrillic	0.211	0.8	0.0068	Hindi	hin_Deva	Devanagari	0.032	0.1	0.0010
Dutch	nld_Latn	Latin	0.071	0.3	0.0065	Hebrew	heb_Hebr	Hebrew	0.025	0.1	0.0009
Romanian	ron_Latn	Latin	0.097	0.4	0.0058	Bengali	ben_Beng	Bengali	0.024	0.1	0.0007
Arabic	arb_Arab	Arabic	0.168	0.7	0.0054	Tamil	tam_Taml	Tamil	0.017	0.1	0.0004

B.5 Evaluation and Analysis Details

BPB computation. To compute BPB, we encode a text separately with the run's tokenizer, prepend a beginning-of-sequence token, and compute the model's loss on every token of the sentence. We then divide by the number of UTF-8 bytes in those sentences. FLORES+ is not part of the model or tokenizer training data. We use the devtest split. Every FLORES+ sentence is shorter than the model's context of 2,048 tokens under every tokenizer, so no truncation occurs and all tokenizers are scored on the same text. We compute Validation BPB analogously, but validation documents may be cropped when they do not fit into the remaining space in a sequence. Because these crop locations can differ across tokenizers, we divide by the number of UTF-8 bytes in the text actually scored by

each model. The three tokenizers whose input is pre-tokenized with the encoding based on Unicode script classes have no exact per-token byte table, so no Validation BPB statistic in this paper includes them.

Seed variation computation. Retraining a model with the same tokenizer from a different random initialization changes its results slightly. We call the size of this change the *seed variation*. For each outcome (BPB, accuracy, etc.), we estimate it as the pooled standard deviation of that outcome across seed replicates of the same tokenizer. The seed changes only the weight initialization; the tokenizer, training data and their order, and hyperparameters are identical across seeds. We do this for five tokenizers: the GPT-4o, Claude, and GPT-4o-NFC balanced BPE tokenizers, the GPT-4o balanced UnigramLM tokenizer, and the GPT-4o SuperBPE tokenizer. For each tokenizer we use only runs trained in the same period, so that differences between runs of one tokenizer come from the random seed alone. For each outcome, we first compute its standard deviation across seed runs within each tokenizer. For per-language outcomes, we do this separately for each tokenizer–language pair. We then pool these standard deviations, weighting each by its degrees of freedom (the number of seed runs minus one). For an outcome with one value per run, the five tokenizers contribute seven degrees of freedom in total. The seed-variation estimates used in the paper differ only in the outcome and in what is pooled:

- *Trained languages*: FLORES+ BPB for each of the 31 trained languages, pooled over the 155 tokenizer–language pairs (5 tokenizers $\times$ 31 languages).
- *Single language*: that language's FLORES+ BPB, pooled over the five tokenizers; these values are shown in Fig. 4.
- *31-language mean*: for each run, we first average FLORES+ BPB over the 31 trained languages, then compute the standard deviation across seeds within each tokenizer and pool over the five tokenizers.
- *Validation BPB*: computed and pooled in the same way over the five tokenizers.

The main analyses use the 31 trained languages; the fixed 214-language FLORES+ set of §2 adds 183 languages that are mostly unseen during training; while we don't explicitly include these languages in our corpora, misidentified or code-switched text from these languages may appear depending on the language filtering. On these languages the run-to-run noise is larger: the per-language seed variation over all 214 languages is 0.0146, about six times the seed variation of 0.0023 over the languages in model-training data.

For the 31 languages of the main analysis, the estimated seed variation is 0.0023 for a single language's FLORES+ BPB, 0.0012 for the 31-language FLORES+ mean, and 0.00041 for validation BPB.

Standard error computation. The 54 tokenizers are not independent observations: they fall into $G = 42$ families, and tokenizers from the same family are likely to behave similarly (§2). The model-based standard errors and p-values from the mixed-effects fits do not account for this dependence and can therefore be too small. We instead use clustered standard errors, estimated from an ordinary-least-squares fit of the same within-language specification with a fixed effect per language and standard errors clustered at the tokenizer-family level ($G = 42$; §2). For the analyses looking at the relationships between intrinsic metrics and downstream outcomes, the OLS coefficients agree with the mixed-effects coefficients to within one unit of the last digit printed in Table 11. Clustering inflates the standard error of each metric by a factor between 1.0 and 3.2 relative to the model-based standard error, with the smallest inflation for the UTF-8 boundary-crossing rate; we therefore rely on the clustered p-values throughout. The tier, log-weight-interaction, and typology fits of §6 use the same construction: an OLS fit with language fixed effects and the metric-by-attribute interaction term, with standard errors clustered by family and BH adjustment within each block of tests. The attribute main effects are absorbed by the language fixed effects.

C Robustness of Design Choices

In this section, we report additional robustness checks for our main conclusions. Specifically, we examine (i) whether the relationships of a language's tokenizer sensitivity and exclusion penalty with its share of the language-model training data hold under alternative ways of measuring them; (ii) whether our results depend on how the outcome is normalized across languages; (iii) whether

resource level can be separated from writing system; (iv) whether the exclusion penalty is confounded by the amount of text a model sees; and (v) whether tokenizer rankings hold beyond repeated data and a single token budget.

Robustness of the BPB spread–model-training data share relationship. The results in §4 suggests that languages with smaller shares of the language-model training data have larger BPB spreads across tokenizers. We test if this result is inflated by how we count tokenizers or by run-to-run seed variation. We first observe that counting near-duplicate tokenizers separately does not create this relationship: when we compute each language's BPB spread across one mean per tokenizer family, using 42 families instead of 54 tokenizers, its Spearman correlation with log language-model training share is -0.54, vs. -0.52 at tokenizer level.

The per-language seed variation also falls as language-model training share rises: the pooled seed standard deviation of a language's FLORES+ BPB correlates with its log training share at Spearman $\rho = -0.608$ ($p = 0.0003$). Because this relationship has the same direction as the BPB-spread relationship, some of the association between BPB spread and training share may reflect larger run-to-run variation for lower-resource languages. Finally, for each language, we take the largest absolute correlation between FLORES+ BPB and any of the nine intrinsic metrics across tokenizers. This value is negatively and significantly correlated with the language's BPB spread (Spearman $\rho = -0.42$): in plain terms, the languages for which tokenizer choice changes BPB the most are not those for which any single intrinsic metric has the strongest relationship with BPB.

Robustness to outcome normalization. Bits per byte (BPB) is not a language-neutral unit (Yang et al., 2026) as it divides the cross entropy loss by the number of UTF-8 bytes in the text the loss is computed over. While FLORES+ is a parallel corpus (i.e. each language subset contains the same content), the number of bytes in the documents can vary greatly across languages. It depends on the number of characters per sentence and on the number of UTF-8 bytes per character of the language's script. For instance, the Chinese sentences have 124 bytes on average, their German translations 159, and their Bengali translations 356, which demonstrates that per-byte information content varies greatly across languages.

For this reason, we also check if dividing by the number of sentences in each document changes the conclusions drawn based on BPB computations. Because the number of bytes per sentence is constant across tokenizers within a language, every within-language comparison is identical under the two outcomes: the tokenizer ranking, the ratio of spread to seed variation, and the sign and significance of each within-language metric coefficient. Cross-language comparisons can differ, but the relationship between tokenizer sensitivity and language-model training share is robust to the choice of normalization. The Spearman correlation between a language's spread across tokenizers and its log share of the language-model training data is -0.52 using the standard deviation of BPB, -0.54 using the standard deviation of bits per sentence, and -0.38 BPB coefficient of variation (when spread is divided by the language's mean BPB); over the 28 languages written with word boundaries, the corresponding correlations are -0.69, -0.69, and -0.51. All are significant.

The identity of the *most* sensitive language does depend on the normalization: Chinese has the largest BPB spread (followed by Bengali and Korean, Fig. 1), but after normalizing BPB variance by the language's mean BPB or when measuring variance in bits per *sentence* rather than bits per *byte*, and Chinese is third, after Bengali and Tamil—the two lowest resource languages.

Similarly, the predictor, a language's share of the language-model training data, is also a byte count; where equal byte counts do not correspond to equal amounts of content across languages (Arnett et al., 2024). As a second measure of a language's share, we divide its model-training bytes by the mean number of bytes of its FLORES+ sentences, which approximates the amount of its training text in FLORES+ sentences. Across the 31 languages, the Spearman correlation between the exclusion penalty and the logarithm of this measure is -0.36 and significant, vs. -0.33 and not significant for the byte share (in Fig. 1, right).

Separating resource level from writing system. The relationship between a language's tokenizer sensitivity and model-training share could be entangled with the underlying writing system. We therefore test if resource level explains a language's tokenizer sensitivity beyond its script. This

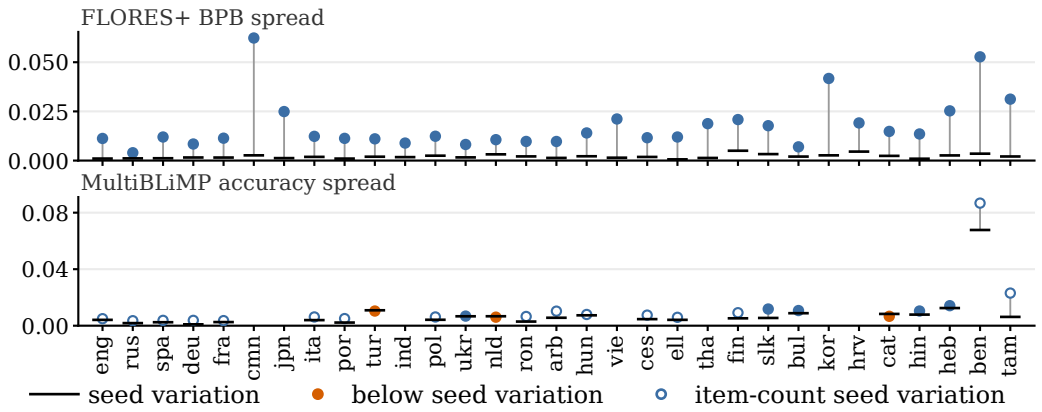

Figure 4: Per-language spread of FLORES+ BPB (top) and MultiBLiMP accuracy (bottom) across the 54 tokenizers, languages in approximately decreasing order of share of the language-model training data (largest at left). Each dot is the standard deviation of the outcome across the 54 tokenizers, and the horizontal tick in the same column, joined to the dot by a thin line, is that language's seed variation; orange dots are languages where the cross-tokenizer standard deviation does not exceed its seed variation (MultiBLiMP: Catalan, Dutch and Turkish). MultiBLiMP accuracy for a language is estimated from a finite number of minimal pairs, so a fixed model's measured accuracy varies with a binomial standard error of $\sqrt{(p(1-p)/n)}$ for n pairs. An open dot marks a language for which the pooled seed standard deviation does not exceed this standard error: for these languages, run-to-run variation cannot be measured more finely than the benchmark's own sampling noise. Languages without MultiBLiMP coverage have no dot in the bottom panel.

analysis uses one observation per language, not the $1{,}674$ tokenizer–language pairs: counting each pair as independent would give false confidence intervals/standard error estimates.

For the across-tokenizer BPB spread of Fig. 1 (where we include one value per language), a non-Latin-script indicator alone explains more of the variance ($R^2 = 0.17$) than logarithm of the model-training data share alone ($R^2 = 0.08$), but neither predictor is stable across specifications: an added indicator for the three languages written without word boundaries is significant, and its addition makes the non-Latin indicator not, and refitting on the 28 languages with word boundaries makes the log share significant (-0.69 vs. -0.52 with all languages included). We report this as a descriptive variance partition, not a causal decomposition: the predictors are entangled in the set of languages we explored. For a different entangled axes, the tokenizer-training share, the resource-inversion experiment of §5 breaks the entanglement by construction: under the inverted-allocation tokenizer, the measured correlation of a language's log share of the tokenizer-training data with its log nominal sampling weight in the language-model training data is -0.961 vs. $+0.199$ and $+0.133$ under the permuted allocations. Given our limited pool of languages, the writing system is not a variable whose effect we can cleanly isolate.

Exclusion penalty confounders. The exclusion penalty result in Finding 2 cannot be attributed cleanly as an effect of tokenizer coverage: excluding a language from the tokenizer-training data also causes the model to see less underlying text from that language in most cases. This occurs because the excluding tokenizers typically segment the language into more tokens, so a fixed training-token budget covers less underlying text. For each of the 30 languages excluded by at least one coverage tokenizer, we compute the ratio of training bytes under the excluding tokenizers to those under the including tokenizers. The ratio is below one for all 30 languages, with a median of 0.82, and languages with smaller ratios have larger exclusion penalties (Spearman $\rho = -0.47$, significant).

No large reordering of tokenizers appears without repeated data or at twice the training budget. Tokenizer rankings across the study come from language model training runs that make two to three passes over the language-model training mixture at a fixed token budget. We test the impact of both design choices on our conclusions. We train 11 additional models with 10 of the tokenizers in the study: one model for each of nine tokenizers from the main set, and two seeds of the inverted-

allocation tokenizer. First, we trained these 11 models on a single-pass corpus at the same token budget. The Spearman correlation between their 31-language FLORES+ means and those of the same tokenizers' trained with our typical repeated-data setup is $+0.93$ ($+0.90$ over ten tokenizers when the two inverted-allocation seeds are averaged first). Second, we continued to train these single-pass run for an additional 10B tokens of new text, to about twice the token budget of our typical setup. The Spearman correlation between the eleven means before and after the continuation is $+0.94$. In both comparisons, the BPB rank agreement is lower for single languages than for the means across languages: the lowest per-language correlations are -0.11 (Chinese) and $+0.23$ (Hindi) between single-pass and repeated data, and $+0.64$ (Finnish) and $+0.83$ (Slovak) across the continuation. With eleven runs, a large reordering would be visible in these comparisons, and a small reordering cannot be distinguished from noise.

D CONTROLLED COMPARISONS

Table 5: Controlled comparisons: 25 tokenizer pairs that differ in one configuration choice. ΔFLORES: variant minus baseline in the 31-language FLORES+ mean. ΔVal: variant minus baseline in validation BPB. The ratio column after each difference is the difference's magnitude divided by the unrounded pooled seed standard deviation of that outcome across seed replicates (§B.5). **Bold**: the difference is larger than run-to-run variation (§2.3), i.e., the ratio is at least $2\sqrt{2} \approx 2.83$; the factor $\sqrt{2}$ converts the standard deviation of one run into the standard deviation of a difference between two runs. [a]full-byte-alphabet rebuild; [b]tokenizer with the byte-drop defect (§B.1)

	Contrast (baseline → variant)	ΔFLORES	ratio	ΔVal	ratio
Data	Balanced → English-only (GPT-4o)[a]	**+0.0386**	32.69	**+0.0332**	81.73
	Balanced → English-only (Claude)[b]	**+0.0307**	26.06	**+0.0248**	61.00
	Balanced → English-only (Punct)[b]	**+0.0238**	20.18	**+0.0254**	62.41
	Balanced → Code-heavy (GPT-4o)	**+0.0189**	16.06	**+0.0153**	37.56
	Balanced → High-resource (GPT-4o)	**+0.0202**	17.09	+0.00025	0.62
	Balanced → High+mid (GPT-4o)	**+0.0121**	10.29	+0.00059	1.45
Algorithm	BPE → UnigramLM (GPT-4o)	**+0.0323**	27.41	**+0.0199**	48.92
	BPE → UnigramLM (Claude)	**+0.0285**	24.19	**+0.0198**	48.81
	BPE → UnigramLM (RightAlign)	**+0.0293**	24.86	**+0.0192**	47.32
	BPE → SuperBPE (GPT-4o)	**+0.0130**	11.02	**+0.0105**	25.94
	PA-BPE → SuperBPE+PA-BPE (GPT-4)	−0.0025	2.11	**+0.0035**	8.51
	BPE → PA-BPE (Clean, NFC)	**+0.0200**	16.95	**+0.0086**	21.11
Pretokenizer	GPT-4o → Claude	+0.0014	1.18	**+0.0024**	5.89
	GPT-4o → GPT-2	−0.0018	1.55	−0.00033	0.81
	GPT-4o → Punctuation	**+0.0048**	4.07	**+0.0054**	13.27
	GPT-4o → RightAlign	+0.0030	2.52	+0.00066	1.62
	GPT-4o → Whitespace	**+0.0049**	4.19	**+0.0078**	19.23
	GPT-4 → Clean (PA-BPE)	**−0.0070**	5.95	**−0.0019**	4.75
NFC	GPT-4o → GPT-4o NFC	−0.0019	1.65	+0.00022	0.55
	Claude → Claude NFC	−0.0036	3.08	−0.00033	0.81
	RightAlign → RightAlign NFC	−0.0038	3.22	+0.00002	0.05
Presets	UnigramLM default → tuned (GPT-4o)	−0.0004	0.33	−0.00002	0.06
	PA-BPE base → hybrid-window (GPT-4)	**−0.0084**	7.11	**−0.0065**	15.88
	SuperBPE+PA-BPE t90k → t64k	−0.0004	0.37	−0.00008	0.20
	SuperBPE Clean C2 → C3	**+0.0044**	3.75	**+0.0012**	2.88

Table 5 presents the results used in §5. Each row compares two tokenizer–model runs whose tokenizers differ in one configuration choice while the language-model architecture, optimization, training mixture, and training-token budget are held fixed. Thus, the comparisons isolate a tokenizer configuration change under our standard training setup. Notably, they do not hold fixed the amount of underlying text seen during language-model training: changing the tokenizer can change compression, so models trained for the same number of tokens may see different numbers of bytes.

The largest change of the 25 comes from restricting the tokenizer-training data to English: it raises both the 31-language FLORES+ mean BPB and validation BPB significantly across three different pretokenization regexes. The milder restrictions, to the high-resource languages or to the high-

and-mid-resource languages, similarly negatively affect performance but to a lesser extent (and not significantly when measured on validation BPB. Of the five pretokenizer-regex changes measured against the GPT-4o baseline, the two largest are the Whitespace pretokenizer (+0.0049 on the 31-language FLORES+ mean) and isolating every punctuation character (+0.0048). NFC normalization lowers the 31-language FLORES+ mean in all three pairs, in two of them (Claude and RightAlign) by more than run-to-run variation. Choice of algorithm also significantly impacts BPB in most cases. BPB consistently increases when switching to the UnigramLM tokenization algorithm. The HuggingFace UnigramLM trainer used here is documented to differ from, and perform worse than, the original SentencePiece implementation, though. Part of the UnigramLM difference may be the implementation rather than the algorithm, and so we do not read strongly into these results.

A byte-alphabet defect lowers measured BPB slightly and does not produce any of the reported BPB increases. A byte-alphabet defect arises when BPE training is run without an explicit initial byte alphabet: every byte value absent from the fitting corpus is then missing from the vocabulary, and text containing such a byte cannot be fully encoded. For the defective English-data tokenizers, 32 of the 128 tested byte values are missing: every C0 control character except tab and newline, the left brace, and the delete character. One pair isolates the defect's impact: the English-data GPT-4o BPE tokenizer of Table 1 is a defect-free rebuild of a defective original, identical in algorithm, pretokenization regex, and tokenizer-training data. The rebuild's 31-language FLORES+ mean is higher than the defective original's by 0.009 and its validation BPB by 0.003; per language, the rebuild's FLORES+ BPB is higher for 30 of the 31 languages, most for Chinese (0.094), and the one difference in the other direction (Bengali) is below the per-language seed variation. The defect therefore ostensibly *lowers* measured BPB. The exclusion penalties of §4 use the rebuild and involve no defective tokenizer, and for the two defective BPE tokenizers retained in the 54-tokenizer panel the defect lowers their measured BPB rather than raising it. We take this as evidence that it does not produce any of the BPB increases this paper reports.

E Equal Weight and Resource-inversion Experiments

In §4, we find that a language's share of language-model training data negatively correlates with the spread of its FLORES+ BPB across tokenizers, i.e., lower-resource languages are more sensitive to tokenizer choice. Most tokenizers in the panel (Table 1) are trained on the *balanced* corpus, where a language's share of the tokenizer-training data is proportional to its share of the language-model training data, so resource-level for language-model and tokenizer training data are entangled. The observed correlation could potentially be an artifact of this. We isolate the impact of tokenizer-training share by reallocating it independently of the language-model training share: we invert the natural allocation, permute it in two ways, exclude languages from it, and weight all languages equally (§E.1). §E.2 reports the correlations between the tokenizer-training data share and changes to FLORES+ BPB under these controlled experiments; §E.3 rules out three alternative explanations for the trends that we observe; and §E.4 compares the results with the exclusion result of §4.

E.1 Set-up

Equal weighting variants For the equal allocation ablations, we train five additional tokenizers where the tokenizer-training weights are equal across 30 non-English languages, and the rate of the English data is held fixed at 35%.

Reference and variant runs for inversion and permutation experiments. We fix the balanced GPT-4o BPE tokenizer and its corresponding model as the reference tokenizer–model pair for the inversion and permutation experiments (§5). We extend our analysis to four additional settings that differ from the reference tokenizer only in the per-language weights of the tokenizer-training data; pretokenizer, normalizer, algorithm, vocabulary size, and special tokens are held constant. We label the four variants by the released artifacts: L2 is trained on inverted-allocation tokenizers, P1 and P2 are the two trained on permuted allocation tokenizers, and E1 is the variant trained without English web text, each elaborated below. We train the L2 model under two random seeds and average the two runs wherever both are available, and we name the seed where an analysis uses only one.

This change affects the exact realized shares (measured in UTF-8 bytes) of the language-model training data. In general, a tokenizer trained on less of a language segments that language into more tokens, so at a fixed number of steps its model is trained on fewer bytes of text. To control for this in these experiments, we therefore set each variant's number of steps so that its model is trained on

the same number of bytes as the reference model: between 9,087 and 9,980 steps vs. 8,800 for the reference model.

Table 6: Equal vs. proportional tokenizer-data weighting across 30 non-English languages (English fixed at 35%). Δ mean: the 31-language FLORES+ mean of the model with equal weights minus that of the model with proportional weights. Δ tail: the same difference in FLORES+ BPB, averaged over the 15 languages with the smallest proportional weights. Spearman ρ: correlation between a language's change in FLORES+ BPB and its proportional weight. Bold and a star mark correlations with $p < 0.05$. Last row: a control that changes only the limit on the number of files.

Variant (vs. balanced baseline)	Δ mean	Δ tail	Spearman ρ
Equal weights (GPT-4o)	$+0.0107$	$+0.0177$	$\mathbf{-0.61}^*$
Equal weights (Claude)	$+0.0064$	$+0.0162$	$\mathbf{-0.52}^*$
Equal weights (Punct)	$+0.0015$	$+0.0031$	-0.07
Equal weights (GPT-4o, NFC)	$+0.0054$	$+0.0099$	-0.34
Equal weights, no repeat sampling	$+0.0060$	$+0.0120$	$\mathbf{-0.44}^*$
File cap only (control)	-0.0035	-0.0039	$+0.17$

E.2 Reallocation Effects

Equal-weighting. We compare each equal-weight tokenizer with the tokenizer trained on the balanced data that matches it in every other configuration choice in Table 6. In every pair the 31-language FLORES+ mean and the mean over the 15 languages with the smallest proportional weights are both *higher* under the equal weights, and in four of the five pairs the rise in the 31-language mean is larger than run-to-run variation.

Coverage tokenizers. In §4, we examine each language's exclusion penalty on a coverage tokenizer trained without it. Table 7 reports the difference in FLORES+ BPB across subsets of the 31 languages.

Inverted and permuted allocations Table 8 reports, for each of the four variants, the Spearman correlation between a language's change in FLORES+ BPB against the reference run and the language's realized log share of the language-model training data. The inverted-allocation tokenizer is the only variant whose correlation is significant, that is, with $p < 0.05$. The difference between the inverted-allocation tokenizer's correlation and each permuted-allocation tokenizer's correlation is tested by a bootstrap, which re-draws the 31 languages with replacement and recomputes both correlations on each draw. The difference is -0.67 against permuted allocation 1 (significant) and -0.30 against permuted allocation 2 (not significant).

E.3 Testing for further confounders

Before attributing Tamil's rise in FLORES+ BPB to the vocabulary itself, we test three alternative explanations: that the inverted-allocation tokenizer simply trains the model on fewer Tamil tokens, that our repeated-data training regime amplifies the effect, and that the rise is concentrated in vocabulary entries the training data contain too rarely to learn well.

Token matching. When a tokenizer segments a language into fewer tokens, the model of that tokenizer is trained on fewer tokens of that language. Under the inverted-allocation tokenizer, the model is trained on fewer Tamil tokens, even though it ends up being trained on more underlying Tamil text. Part of the increase we observe could be due to this phenomenon rather than an effect of the vocabulary itself.

To test this, we train two additional models on a reweighted language-model training mixture, one with the reference tokenizer (BPE GPT-4o tokenizer trained on balanced mixture) and one with the inverted-allocation tokenizer. On this reweighted mixture, a model with the inverted-allocation tokenizer is trained on as many tokens of each language as the reference model is on the original mixture. Matching token counts this way means the model on the reweighted mixture is trained on a *different amount of underlying text*, compared to the set-up described above. Table 9 reports the four combinations of tokenizer (reference or inverted) and language-model training mixture (original, or reweighted so that the model with the inverted-allocation tokenizer is trained on as many tokens of each language as the reference model is on the original mixture; §5).

Table 7: The difference between the coverage tokenizers that is not specific to a language, for the three groups of languages that the same coverage tokenizers exclude. Languages: the number of languages in the group. Difference in 31-language mean: the average of the 31-language FLORES+ means of the coverage tokenizers that exclude the group's languages, minus the average of the 31-language FLORES+ means of the coverage tokenizers that include them. Mean penalty: the mean of the exclusion penalties (§4) of the group's languages. Ratio: the difference in 31-language mean divided by the mean penalty. English is not included in the analysis below.

# of coverage tokenizers that exclude the language	**# of languages in group**	**Difference in 31-language mean**	**Mean penalty**	**Ratio**
One (English-only)	5	0.0278	0.0671	0.41
Two (English-only, high-resource)	15	0.0233	0.0213	1.09
Three (English-only, high-resource, high-and-mid-resource)	10	0.0236	0.0330	0.72

Table 8: Spearman correlation, for each reallocated tokenizer, between a language's change in FLORES+ BPB against the reference run and the language's log share of the language-model training data (realized share), over the 31 trained languages. A star marks $p < 0.05$.

Label	Variant	Spearman ρ
L2	inverted-allocation tokenizer (two seeds averaged)	-0.59^*
P1	permuted allocation 1	$+0.08$
P2	permuted allocation 2	-0.28
E1	no English web text	-0.13

On the reweighted mixture, the model with the inverted-allocation tokenizer is $+0.181$ above the reference model, and $+0.233$ above the model with the reference tokenizer trained on the same reweighted mixture ($0.181 - (-0.052)$); this between-tokenizer difference at matched Tamil token counts is close to the $+0.267$ difference at matched steps on the original mixture. This suggests that with repeated data, the smaller number of Tamil training tokens is not the main cause of Tamil's higher FLORES+ BPB.

Table 9: Tamil's FLORES+ BPB minus that of the reference model on the original mixture, for the four combinations of tokenizer and language-model training mixture. Reweighted mixture: the mixture on which the model with the inverted-allocation tokenizer is trained on as many tokens of each language as the reference model is on the original mixture.

	Original mixture	Reweighted mixture
Reference tokenizer	0	-0.052
Inverted-allocation tokenizer	$+0.267$	$+0.181$

Single-pass replication. Every language-model training run of the 54 tokenizers and every additional run described so far in this section makes between two and three passes over the language-model training mixture (§2). To test whether repeating training data during language-model training matters, we train three additional models on a larger mixture of the same composition in which no document is processed twice: one run of the reference tokenizer and two seeds of the inverted-allocation tokenizer, with final checkpoint scores for the two averaged. We contruct this single-pass mixture so that the data seen in the beginning matches the original run's first pass, document for document; the two mixtures differ only after the point at which the original run begins its second pass. Before that point, a single-pass run and its repeated-data counterpart have the same tokenizer, data, and seed. As a check that the two runs match until that point, we compare, at steps $1{,}000$ and $2{,}500$, the per-language change in FLORES+ BPB under the inverted-allocation tokenizer on single-pass data with the same change on repeated data (seed 42). Of the 62 language-step entries, 4 differ by more than the pre-set tolerance of 0.02, and the largest difference is 0.0221 (Turkish at step $1{,}000$). We attribute these differences to evaluation at single checkpoints.

Table 10 reports the per-language change in FLORES+ BPB under the inverted-allocation tokenizer against the reference run, on repeated data and on single-pass data. Tamil's change is $+0.1386$ on single-pass data against $+0.2674$ on repeated data. The pre-registered criterion for a material effect of repetition was that the single-pass change falls below 0.7 of the repeated-data change; the criterion is met. For the two other languages of Table 10, the change on single-pass data is within 0.01 of zero.

The Spearman correlation between a language's change in FLORES+ BPB and its log share of the language-model training data is -0.59 on repeated data and $+0.34$ on single-pass data. The repeated-data correlation is significant; the single-pass correlation is not. Computed separately for the two single-pass seeds, the correlation is $+0.33$ and $+0.30$, where neither is significant. The sign of the single-pass correlation does not depend on any one language: leaving out each of the 31 languages in turn and recomputing the correlation ranges from $+0.29$ to $+0.47$. Using all 31 languages, the correlation is not significant, and it stays not significant when 26 of the 31 languages are held out in turn. Chinese has the largest single-pass rise ($+0.14$); leaving it out lowers the correlation to $+0.29$ with $p = 0.12$.

For each language, we compute the change on repeated data minus the change on single-pass data, to quantify how much repetition hurts performance in that language. We find that this difference correlates with the language's log share of the language-model training data at Spearman -0.89 (significant): *repetition hurts most for Bengali and Tamil, the two languages whose fertility ratio fell furthest under the inverted-allocation tokenizer*. Under the reference tokenizer, the single-pass run's 31-language FLORES+ mean is lower than the repeated-data run's by 0.0164, so *single-pass data lowers BPB in general* independent of the change in allocation. The per-language difference above removes this general shift, isolating the language-specific effect of repetition. The results of §5 are therefore results for repeated-data training, which every run of the 54 tokenizers shares. On single-pass data, about half of Tamil's rise remains, and the rise is not larger for the languages with a smaller share of the language-model training data.

Table 10: Per-language change in FLORES+ BPB from the reference run under the inverted-allocation tokenizer, on the repeated training data of the 54 tokenizers' runs and on single-pass data: the two languages whose fertility ratio fell furthest, two languages whose fertility ratio changed less, and the mean over the 31 training languages. In both settings the inverted-allocation value is the average of two seeds, compared against one reference run; the largest difference between the two seeds in any language's change is 0.017 on repeated data and 0.019 on single-pass data.

	Repeated	**Fresh (single epoch)**
Tamil	$+0.2674$	$+0.1386$
Bengali	$+0.2616$	$+0.0997$
Hebrew	$+0.0724$	0.0059
Korean	$+0.0468$	-0.0087
Mean (31 trained languages)	$+0.0256$	$+0.0123$

Vocabulary entries that the language-model training data contains rarely. One candidate explanation for Tamil's rise is that the inverted-allocation tokenizer adds vocabulary entries for Tamil that occur too rarely in the language-model training data, which could leave their embeddings poorly trained. To test this, we estimate, how often each vocabulary entry occurs in a run's training-token stream of one inverted-allocation seed, then pair each per-token loss from that run's FLORES+ evaluation with its entry's training-frequency count. Under the inverted-allocation tokenizer, 2.6% of Tamil's FLORES+ token occurrences fall on entries that the training stream contains ten or fewer times vs. 0.0% under the reference tokenizer. These rare entries are indeed harder to predict: among Tamil entries of 15 to 19 bytes seen 100 or fewer times cost 1.13 bits per byte vs. 0.54 for entries of the same length seen more than $1{,}000$ times. But these entries do not account for Tamil's rise: only 3.9% of the run's Tamil bits fall on them, and crediting them with every bit they receive accounts for at most 13.6% of the run's Tamil rise of $+0.277$. The rise is instead spread over the vocabulary: entries that are the same string under both tokenizers, with training-stream counts matched within a factor of 1.25, cost 1.61 times as many bits in Tamil under the inverted-allocation tokenizer as under the reference tokenizer.

E.4 Exclusion and reallocation compared

Excluding a language from the tokenizer-training data (§4) and raising its share (§5) change the same quantity in opposite directions, and both raise the language's FLORES+ BPB. When a language is excluded, its text is segmented into more tokens, and its penalty grows with its fertility ratio. When a language's share is raised, its text is segmented into fewer tokens, and the rise does not come from a smaller byte budget: under the inverted-allocation tokenizer, Tamil's share of the seed-42 model's training bytes rose by 0.015 percentage points while its FLORES+ BPB rose by $+0.267$ (mean of two seeds). Of Tamil's rise on repeated data, matching its training-token count removes about a third (Table 9), and about half appears only with repeated data (Table 10).

F Intrinsic Metrics: Definitions and Additional Results

F.1 Intrinsic metric definitions

Each metric is computed from a tokenizer applied to a text corpus, without training a model; we measure each on the same FLORES+ text per language (and, in §F.2, on FineWeb2 text). All values are computed with the TokEval library; each metric's exact definition is stated below.

- **Compression rate**: UTF-8 bytes of text per token. Higher means each token covers more text.
- **Fertility**: tokens per whitespace-delimited word. Higher means words are split into more pieces. Fertility is closely related to compression rate (the two are strongly correlated across tokenizers) and sits between an information-theoretic measure of compression and a morphological measure of how words are segmented.
- **Unigram, bigram, and trigram entropy**: the Shannon entropy, in bits, of the distribution of single tokens, token pairs, and token triples in the tokenized corpus. Lower entropy means a more peaked, more predictable token distribution.
- **Rényi efficiency** ($\alpha = 2$): the order-2 Rényi entropy of the unigram token distribution, normalized by $\log_2$ of the declared vocabulary size, so that it lies in $[0, 1]$. Relative to Shannon entropy it down-weights the long tail of rare tokens, and it was proposed as a predictor of language-model quality (Zouhar et al., 2023). Values normalized instead by the number of observed token types are on a different, per-language-incomparable scale and cannot be compared with the values here.
- **Vocabulary utilization**: the fraction of the vocabulary the tokenizer actually uses on the corpus. Because it is the used-token count divided by vocabulary size, it is not comparable across tokenizers of different vocabulary sizes. One further caution for reuse: under the compression-rate covariate control its within-language coefficient jumps from $+0.0002$ to $+0.0105$ per standard deviation of the metric, a shift far larger than any other metric's, so the controlled coefficient is a collinearity artifact with compression rate and we do not interpret it.
- **UTF-8 character split rate**: among the multi-byte UTF-8 characters in the text, the fraction the tokenizer splits across a token boundary, so that one character is spread over more than one token. It is well-defined only where multi-byte characters are common.
- **UTF-8 boundary-crossing rate**: the fraction of tokens whose byte span crosses a UTF-8 character boundary (a token that begins or ends in the middle of a character). Both UTF-8 rates are detected at the byte level as defined here.

F.2 Per-language results on in-distribution FineWeb2 text and the full FLORES+ language set

The main analyses correlate intrinsic metrics measured on FLORES+ with BPB measured on FLORES+, which is parallel, curated text outside the training distribution. This analysis looks at how results change when using in-distribution text for both measurements. Here, the intrinsic metrics are computed on FineWeb2 text, and the outcome is the per-language FineWeb2-validation BPB, bits-per-byte on 100 held-out FineWeb2 documents. The section further looks at how results change when considering languages not explicitly in the model's training data mixture, i.e., when considering the full FLORES+ language set. These values are computed across all 54 tokenizers over the 31 trained languages.

The within-language relationships hold in-distribution and are generally stronger. Table 12 repeats the within-language mixed effects model of Table 11 using FineWeb2 data. The coefficients of eight of the nine metrics have the same sign as in the FLORES+ analysis, and seven of these eight are larger in magnitude in-distribution: for example, fertility's coefficient is $+0.0322$ on FineWeb2 vs. $+0.0156$ on FLORES+. Further, all nine metrics are significant when using the family-clustered

Table 11: Within-language relationship between intrinsic metrics and FLORES BPB on the 31 trained languages: β from BPB $\sim z(m) + (1 \mid \text{language})$, in BPB per standard deviation of the metric ($n = 1674$ tokenizer–language pairs; 1673 for UTF-8 char split, whose value is unavailable for one (tokenizer, language) entry; 31 languages, 54 tokenizers; null-model language ICC $= 0.995$). SE_m: model-based standard error. SE_cl and p^cl_adj: cluster-robust standard error and BH-adjusted p-value, clustered at the family level ($G = 42$ families); the clustered p is the inference we rely on. R^2: share of within-language variance explained. Sorted by $|\beta|$. Vocabulary utilization's raw R^2 is marginally negative, a degenerate output reported as 0.000. All values are for models trained on the same number of tokens; Table 14 reports the coefficients after the adjustment for the number of training bytes.

Intrinsic metric	β	SE_m	SE_cl	p^cl_adj	R^2
Fertility	$+0.0156$	0.0013	0.0027	$< 10^{-7}$	0.082
UTF-8 char split	$+0.0113$	0.0005	0.0012	$< 10^{-16}$	0.214
Trigram entropy	-0.0112	0.0006	0.0015	$< 10^{-13}$	0.170
Bigram entropy	-0.0083	0.0006	0.0017	$< 10^{-5}$	0.120
Compression rate	-0.0072	0.0007	0.0024	0.003	0.057
Rényi eff. (α=2)	-0.0070	0.0005	0.0015	$< 10^{-5}$	0.093
Unigram entropy	-0.0064	0.0006	0.0021	0.003	0.056
UTF-8 boundary crossing	$+0.0028$	0.0008	0.0008	$< 10^{-3}$	0.007
Vocab utilization	$+0.0013$	0.0010	0.0030	0.67	0.000

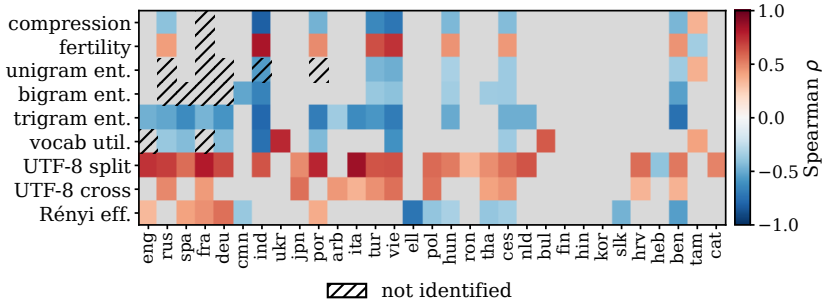

Figure 5: Per-language Spearman ρ between each intrinsic metric and FLORES+ BPB across the 54 tokenizers, for the 31 languages of the training mixture ordered from the highest to the lowest share of it. Only BH-FDR-significant entries are colored. The boundary-crossing entries for English, Indonesian, and Dutch are undefined (zero variance), and the char-split entry for Korean has $n = 53$. Hatching marks the 13 entries that are not identified under the measured-composition control of §F, where the metric and the measured number of training bytes of the language correlate at $|\rho| > 0.9$ across the tokenizer set.

standard error estimates on FineWeb2, vs. eight of nine on FLORES+. Vocabulary utilization is the exception: it is not a significant predictor of FLORES+ BPB, but on FineWeb2 its coefficient is -0.0076 and significant, so tokenizers that use more of their vocabulary for a language have lower in-distribution BPB on that language.

Other per-language results also hold in-distribution. Of the 279 metric-by-language correlations, 211 are significant on FineWeb2 against 104 on FLORES+ (Fig. 6). The correlation between a language's BPB spread across tokenizers and its log training-data weight remains negative and significant (-0.394 on FineWeb2 against -0.541 on FLORES+). The mean off-diagonal Kendall τ between per-language tokenizer rankings is 0.455 on FineWeb2 against 0.418 on FLORES+ (§3), showing that disagreement between language-specific rankings is not specific to FLORES+; Ukrainian's agreement rises to 0.309 and is no longer the lowest, suggesting that its extreme FLORES+ value is specific to that text.

Results on entire FLORES+ dataset. At the aggregate level, for most metrics the sign of the correlation between the tokenizer's global metric value and its mean per-language FLORES+ BPB flips when the mean is taken over all 214 FLORES+ languages instead of the 31 trained languages (Table 13); Rényi efficiency is the exception. There are several possible reasons that could explain

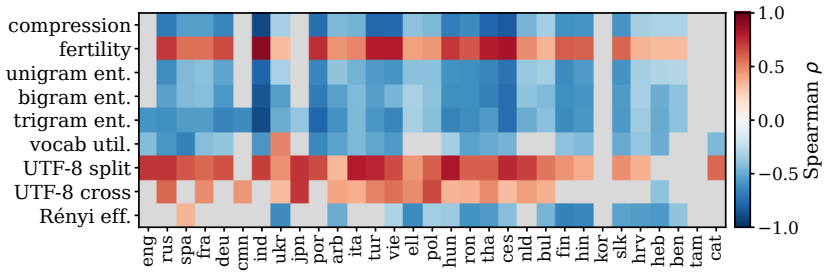

Figure 6: Per-language Spearman ρ between each FineWeb2-measured intrinsic metric and per-language FineWeb2-validation BPB, across the 54 tokenizers, for the 31 trained languages ordered from highest to lowest training-data share. FineWeb2 analogue of Fig. 5.

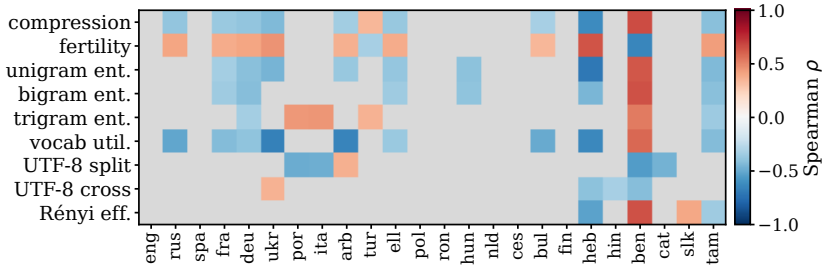

Figure 7: Per-language Spearman ρ between each intrinsic metric (measured on FLORES+ text, as in Fig. 5) and per-language MultiBLiMP accuracy, across the 54 tokenizers, for the 24 covered trained languages ordered from highest to lowest training-data share. The boundary-crossing entries for English and Dutch are undefined (zero variance).

this effect. The 214-language mean is a mean over 31 trained and 183 untrained languages, so the untrained languages contribute most of its terms. As the median columns of Table 13 show, the per-language correlations between a tokenizer's global metric value and one language's FLORES+ BPB have, for most metrics, one sign on the trained languages and the opposite sign on the untrained languages; the aggregate correlation follows whichever set the mean is taken over, and no single language's relationship changes. Rényi efficiency is the exception: its aggregate correlation keeps its sign on both language sets. The direction of the change is consistent with the finite-vocabulary trade-off that motivates this study: the tokenizers with the best global metric values on the trained mixture are those whose vocabularies fit the trained languages most closely, and §4 shows that a language absent from the tokenizer-training data has higher BPB. The 183 untrained languages are absent from both the tokenizer-training and the language-model training data. Two cautions apply. The untrained-language correlations use the tokenizer's global metric value, not a measurement on the untrained language's own text. And the per-language seed variation over the 214 languages is 0.0146, about six times the 0.0023 of the trained languages (§B.5), so the individual untrained-language correlations are noisier than their trained-language counterparts.

F.3 Controlling for confounders

The training budget is fixed in tokens, so a model whose tokenizer compresses a language better is trained on more bytes of that language. For a fixed token budget, several metrics (e.g., compression rate and fertility) are therefore directly related to how much underlying text a model will see. This could be a confounder in any relationship we observe between one of these metrics and BPB. We measure each run's number of training bytes of each language with a census of the training data for 51 of the 54 tokenizers. The three tokenizers that use the encoding based on Unicode script classes (§2) train on pre-tokenized shards, for which we did not record the number of bytes; they are excluded from both analyses below. We use the measurement as an offset to the pooled within-language coefficients of Table 11, and as a covariate for the per-language correlations of Fig. 5.

Pooled coefficients adjusted for training bytes. Let x be the natural logarithm of a run's number of training bytes of the language, $\bar{x}$ its mean, and β an estimate of the change in BPB per unit increase

Table 12: Within-language relationship between intrinsic metrics and per-language FineWeb2-validation BPB on the 31 trained languages, with both sides measured on FineWeb2 text: β from BPB $\sim z(m) + (1 \mid \text{language})$, in BPB per standard deviation of the metric ($n = 1674$ tokenizer–language pairs; 31 languages, 54 tokenizers; null-model language ICC $= 0.994$). Columns as in Table 11: model-based and family-clustered standard errors ($G = 42$), BH-adjusted clustered p-value, share of within-language variance explained. Sorted by $|\beta|$.

Intrinsic metric	β	SE_m	SE_cl	$p^{\text{cl}}_{\text{adj}}$	R^2
Fertility	$+0.0322$	0.0014	0.0022	$< 10^{-47}$	0.244
Trigram entropy	-0.0168	0.0005	0.0016	$< 10^{-26}$	0.458
Unigram entropy	-0.0151	0.0005	0.0020	$< 10^{-13}$	0.329
Compression rate	-0.0151	0.0007	0.0027	$< 10^{-7}$	0.240
Bigram entropy	-0.0145	0.0004	0.0017	$< 10^{-17}$	0.396
UTF-8 char split	$+0.0140$	0.0005	0.0008	$< 10^{-71}$	0.344
Rényi eff. (α=2)	-0.0121	0.0005	0.0015	$< 10^{-14}$	0.298
Vocab utilization	-0.0076	0.0010	0.0008	$< 10^{-18}$	0.035
UTF-8 boundary crossing	$+0.0022$	0.0007	0.0010	0.026	0.005

Table 13: Per-language decomposition of the full FLORES+ language set, across the 54 tokenizers. Agg. ρ: Spearman between the tokenizer's global metric value and its mean per-language FLORES+ BPB over the 31 trained languages (tr.) and over all 214 languages. Median lang. ρ: median of the per-language correlations (global metric value against one language's BPB) over the trained and the untrained languages. Sig.: languages with a significant per-language correlation (BH-FDR within metric), of 31 trained and of 183 untrained.

	Agg. ρ		**Median lang. ρ**		**Sig.**	
Intrinsic metric	**tr.-31**	**all-214**	**trained**	**untrained**	**tr.**	**untr.**
Rényi eff. (α=2)	-0.455	-0.193	-0.288	-0.072	9	11
Trigram entropy	-0.446	$+0.130$	-0.328	$+0.080$	15	34
Compression rate	-0.326	$+0.227$	-0.226	$+0.106$	9	56
UTF-8 char split	$+0.331$	-0.194	$+0.348$	-0.047	17	34
Bigram entropy	-0.286	$+0.052$	-0.214	$+0.024$	6	40
Unigram entropy	-0.274	$+0.100$	-0.224	$+0.068$	5	50
UTF-8 boundary crossing	$+0.266$	-0.126	$+0.273$	0.005	5	10
Vocab utilization	-0.040	$+0.245$	-0.104	$+0.079$	2	33
Fertility	$+0.033$	-0.303	$+0.008$	-0.125	1	28

in x. We replace each per-language BPB of a run by BPB $- \beta\,(x - \bar{x})$, with the same β for every run and every language, and refit the within-language models of §2 and the tier fits of §6. We do not fit β in these models; instead we take it from runs that differ from their counterparts only in the number of training bytes. Several of these runs exist from various ablations performed throughout the study: the byte-matched retrains of the SuperBPE and UnigramLM tokenizers, the thirteen repeated-data continuations, and the eleven single-pass continuations. This gives four estimates of β, the largest about twice the smallest, and Table 14 reports the refit under each. The adjustment does not change the sign of any coefficient. Fertility, the UTF-8 character-split rate, bigram and trigram entropy, and the UTF-8 boundary-crossing rate remain significant under all four estimates. Compression rate is significant under only the smallest estimate, and unigram entropy and Rényi efficiency lose significance under the larger estimates. The relationships that remain are those of metrics that describe how text is segmented; compression rate, the metric most directly tied to the number of training bytes, does not retain a within-language relationship with BPB once that number is adjusted for. The tier fits of Table 15 change in the same way.

Per-language correlations given the language's training bytes. For each entry of Fig. 5, we recompute the Spearman correlation between the metric and the language's BPB as a partial correlation given the run's number of training bytes of the language, over the 51 tokenizers. An entry is not identified when the metric and the number of training bytes correlate at $|\rho| > 0.9$ across the tokenizers, because the partial correlation cannot then separate them. Thirteen of the 276 entries are not identified, and they are hatched in Fig. 5. Of the identified entries that are significant at tokenizer

Table 14: Within-language coefficients of Table 11 after each run's BPB is adjusted for its number of training bytes (§F.3), for the 51 tokenizers with a measured number of training bytes. In each column the adjustment is made with one estimate of β, the change in the 31-language FLORES+ mean per unit increase in the natural logarithm of the number of training bytes: from left to right, no adjustment, the SuperBPE byte-matched retrain, the thirteen repeated-data continuations, the UnigramLM byte-matched retrain, and the eleven single-pass continuations. *: BH-adjusted family-clustered $p < 0.05$. The unadjusted column is the fit of Table 11 on the same 51 tokenizers, so its values differ slightly from that table, which uses all 54.

	unadjusted	**estimate of β used in the adjustment**			
Metric		-0.0186	-0.0257	-0.0365	-0.0397
Fertility	$+0.0154^*$	$+0.0154^*$	$+0.0147^*$	$+0.0135^*$	$+0.0135^*$
UTF-8 char split	$+0.0113^*$	$+0.0113^*$	$+0.0108^*$	$+0.0099^*$	$+0.0099^*$
Trigram entropy	-0.0110^*	-0.0109^*	-0.0095^*	-0.0070^*	-0.0072^*
Bigram entropy	-0.0083^*	-0.0082^*	-0.0067^*	-0.0039^*	-0.0041^*
Compression rate	-0.0071^*	-0.0070^*	-0.0055	-0.0029	-0.0031
Rényi eff. (α=2)	-0.0071^*	-0.0070^*	-0.0055^*	-0.0027^*	-0.0029
Unigram entropy	-0.0065^*	-0.0064^*	-0.0050^*	-0.0026	-0.0028
UTF-8 boundary crossing	$+0.0025^*$	$+0.0025^*$	$+0.0023^*$	$+0.0020^*$	$+0.0020^*$
Vocab. utilization	$+0.0011$	$+0.0012$	$+0.0022$	$+0.0041$	$+0.0040$

Table 15: Within-language coefficient β of each intrinsic metric on FLORES+ BPB, fit separately within resource tiers (high n=6, mid n=15, low n=10 languages, by share of the language-model training data), in BPB per standard deviation of the metric. Sorted by the low-tier $|\beta|$. The coefficients are the mixed-effects tier fits. **Bold**: BH-adjusted $p_{\text{adj}} < 0.05$ under the family-clustered tier fit, an OLS fit with language fixed effects clustered by family ($G = 42$; §B.5); the compression-rate high-tier entry is not significant ($p_{\text{adj}} = 0.063$), and it is also not significant after the adjustment for the number of training bytes (§F.3).

Intrinsic metric	**High**	**Mid**	**Low**
Trigram entropy	$\mathbf{-0.0111}$	$\mathbf{-0.0084}$	$\mathbf{-0.0128}$
UTF-8 char split	$\mathbf{+0.0215}$	$\mathbf{+0.0074}$	$\mathbf{+0.0119}$
Vocab utilization	-0.0052	$+0.0012$	$\mathbf{+0.0112}$
Rényi eff. (α=2)	-0.0002	$\mathbf{-0.0065}$	$\mathbf{-0.0105}$
Fertility	$\mathbf{+0.0539}$	$\mathbf{+0.0070}$	$\mathbf{+0.0099}$
Bigram entropy	-0.0060	$\mathbf{-0.0077}$	$\mathbf{-0.0094}$
Unigram entropy	-0.0062	$\mathbf{-0.0059}$	-0.0059
Compression rate	-0.0103	$\mathbf{-0.0063}$	-0.0043
UTF-8 boundary crossing	$\mathbf{+0.0101}$	$\mathbf{+0.0028}$	$+0.0028$

level, about three-fifths remain significant under the token count partial correlation, and about one in ten change sign. The sign changes are concentrated in specific languages: four are Thai entries, three Hungarian, two Chinese, one Arabic, and one Czech, and nine of the eleven are entries of the entropy metrics or of Rényi efficiency. The entries that lose significance also concentrate by metric: one of unigram entropy's six significant entries remains significant under the partial correlation, two of bigram entropy's eight, sixteen of the character-split rate's twenty-one, and twelve of trigram entropy's fifteen. The reasons behind the sensitivity of these measurements is an investigation we leave for future work.

Rényi efficiency is not significantly related to FLORES+ BPB for the languages with the most language-model training data. We ask whether a metric's within-language coefficient depends on the language's share of the language-model training data. We order the 31 languages by this share, form groups of high-resource, mid-resource, and low-resource languages, and refit the within-language model within each group (Table 15 in §F). The coefficient of Rényi efficiency is -0.0009 and not significant in the high-resource group, and -0.0116 and significant in the low-resource group.[5] We also fit one model per metric on all 31 languages, with an interaction between the

[5]We standardize each metric within its group before the fit, so the magnitudes of the coefficients are not directly comparable between groups.

Table 16: Pairwise accuracy at ordering held-out tokenizer pairs by the 1.27B models' aggregate BPB, under leave-one-family-out cross-validation over the 42 tokenizer families (40 for the validation-BPB target, which excludes the three SCRIPT tokenizers; §B.5). Val BPB: validation BPB. FLORES-tr.: the 31-language FLORES+ mean. Tokenizer-level rows (second block) use one value of each metric per tokenizer. Per-language rows (third block) use the same metrics measured on each of the 31 languages (278 features; §F.4 states the excluded column); a single-metric row of that block uses all per-language measurements of that metric as the features. The majority-class baseline is the accuracy of always predicting the more frequent pair order. Each single-metric row uses one metric fixed in advance; selecting the best single-metric row from this table afterward is a choice made in hindsight, so we recommend the fixed nine-metric combination for screening. The three smaller-model rows are direct ranking transfer, not cross-validation: the 17 tokenizers with runs at all four scales are ranked by the smaller model's validation BPB and the fraction of pairs ordered as by the 1.27B outcome is reported.

Ranking method	**Val BPB**	**FLORES-tr.**
Majority-class baseline	0.601	0.565
All nine metrics (logistic ridge)	0.700	0.735
Best single metric, selected per fold	0.655	0.668
Bigram entropy (fixed)	0.683	0.676
Trigram entropy (fixed)	0.639	0.664
Rényi eff. (α=2, fixed)	0.675	0.653
Compression rate (fixed)	0.669	0.657
All nine metrics, per language (278 features)	0.907	0.890
Best single feature, selected per fold	0.848	0.799
Bigram entropy block (fixed)	0.910	0.909
Trigram entropy block (fixed)	0.879	0.865
Rényi eff. block (α=2, fixed)	0.860	0.824
Compression rate block (fixed)	0.828	0.832
Smaller model, d8 (val-BPB rank, n=17)	0.868	0.875
Smaller model, d12 (val-BPB rank, n=17)	0.846	0.853
Smaller model, d16 (val-BPB rank, n=17)	0.868	0.831

metric and the logarithm of the language's share. For Rényi efficiency the interaction is significant ($+0.0061$), so its negative coefficient is closer to zero for languages with a larger share. For the character-split rate the interaction is also significant ($+0.0165$), so its positive coefficient is larger for languages with a larger share. For the other seven metrics, fertility among them, the interaction is not significant.

F.4 Tokenizer ranking

Implementation. To predict which of two tokenizers will have better downstream BPB, we train logistic pairwise rankers. For each pair of tokenizers (a, b) in a fixed canonical order, the feature vector is the difference of the two tokenizers' metric vectors, and the label is whether b's aggregate BPB exceeds a's. Features are z-scored (standardized to mean zero and standard deviation one) using statistics computed on the training pairs only. The classifier is scikit-learn `LogisticRegression` with an L2 penalty, the `lbfgs` solver, and an intercept; the prediction for a held-out pair is the sign of the fitted linear score. The ridge strength C is selected inside each training fold by an inner tokenizer family-wise cross-validation over a 13-point logarithmic grid from 10^{-3} to 10^{3}. The procedure is deterministic and uses no random seeds. The tokenizer-level feature space has one value of each of the nine metrics per tokenizer. The per-language feature space has the same nine metrics measured on each of the 31 languages: $9 \times 31 = 279$ columns, minus the Korean character-split column, which is unavailable for one tokenizer (Fig. 5), leaving 278. To account for dependence among closely related tokenizers, we use leave-one-family-out cross-validation. In each fold, we hold out an entire tokenizer family, so that a tokenizer is never ranked by a model trained on another member of its own family. The 31-language FLORES+ mean target uses all 54 tokenizers (42 families); the validation-BPB target excludes the three SCRIPT tokenizers, for which no validation BPB exists (§B.5), leaving 51 tokenizers in 40 families. We evaluate pairwise accuracy, the fraction of evaluated tokenizer pairs whose predicted ordering matches their ordering according to the trained models' BPB. We score every pair for which at least one tokenizer belongs to the held-out family.

Per-language ranking accuracies. For each of the 31 languages, we rank the 54 tokenizers by that language's FLORES+ BPB and score the ridge combination of the nine metrics on held-out pairs under the same leave-one-family-out folds as Table 16. In the per-language feature space, the nine columns measured on the target language itself are dropped before fitting, so a language's ranking is never predicted from metrics measured on its own text. With the metrics as one value per tokenizer, the per-language accuracies have mean 0.713 and median 0.709 and range from 0.615 (Croatian) to 0.802 (Tamil) vs. 0.735 for the 31-language FLORES+ mean target. With the metrics measured per language, the accuracies have mean 0.829 and median 0.837 and range from 0.738 (Czech) to 0.878 (Tamil) vs. 0.890 for the same target.

G TASK ACCURACIES AS THE OUTCOME: MULTIBLIMP AND XCOPA

We repeat the within-language analyses with per-language MultiBLiMP and XCOPA accuracy in place of BPB.

MultiBLiMP. MultiBLiMP accuracy (§2) is the fraction of minimal pairs whose grammatical member the model assigns higher probability. It is available for 24 of the 31 trained languages and all 54 tokenizers. Per-language accuracies range from 0.667 to 1.0; both endpoints are Bengali, which has only 21 minimal pairs and therefore the noisiest estimates in our language set. For 16 of the 24 languages, the pooled seed standard deviation does not exceed the binomial standard error implied by the benchmark item count, so the uncertainty from the finite benchmark is at least as large as the observed run-to-run variation; these are the open dots in Fig. 4.

Table 17 repeats the within-language model of §2 with MultiBLiMP accuracy as the outcome. Eight of the nine intrinsic metrics are significant under the family-clustered inference, with vocabulary utilization again the exception. At the individual-language level, 67 of the 216 metric-by-language correlations are significant after adjustment across all entries (Fig. 7).

Table 17: Within-language relationship between intrinsic metrics and per-language MultiBLiMP accuracy on the 24 covered trained languages: β from accuracy $\sim z(m) + (1 \mid \text{language})$, in accuracy per standard deviation of the metric ($n = 1296$ tokenizer–language pairs; 24 languages, 54 tokenizers). SE_m, SE_cl, and $p^{\text{cl}}_{\text{adj}}$ as in Table 11 ($G = 42$ families; the clustered p-value is the inference we rely on). Sorted by $|\beta|$.

Intrinsic metric	β	SE_m	SE_cl	$p^{\text{cl}}_{\text{adj}}$
UTF-8 char split	-0.0066	0.0006	0.0014	$< 10^{-5}$
Fertility	-0.0056	0.0007	0.0012	$< 10^{-5}$
UTF-8 boundary crossing	-0.0046	0.0009	0.0009	$< 10^{-5}$
Trigram entropy	$+0.0040$	0.0007	0.0010	$< 10^{-4}$
Bigram entropy	$+0.0035$	0.0006	0.0009	$< 10^{-3}$
Rényi eff. (α=2)	$+0.0031$	0.0006	0.0007	$< 10^{-4}$
Unigram entropy	$+0.0029$	0.0007	0.0009	0.003
Compression rate	$+0.0029$	0.0008	0.0009	0.004
Vocab utilization	-0.0002	0.0011	0.0010	0.64

XCOPA. XCOPA (Ponti et al., 2020) is a two-alternative causal-commonsense benchmark scored by loglikelihood, with chance accuracy 0.5. It covers seven of the 31 trained languages, with 500 items per language, and we evaluate all 54 tokenizers and the five allocation runs of §E. Across the 54 tokenizers, per-language accuracies span only 0.468 to 0.586. Nevertheless, mean XCOPA accuracy is associated with the BPB outcomes: it correlates with validation BPB at Spearman $\rho = -0.49$ ($n = 51$) and with the 31-language FLORES+ mean at $\rho = -0.52$ ($n = 54$); both correlations are significant.

The per-language intrinsic-metric relationships are much weaker on XCOPA: only 3 of the 63 metric-by-language correlations are significant after BH adjustment, and no metric is significant in the family-clustered within-language model ($G = 42$ families). Five of the seven languages have a cross-tokenizer accuracy spread larger than their seed variation. For all seven, however, the pooled seed standard deviation is no larger than the binomial standard error at 500 items, so benchmark sampling uncertainty is at least as large as the observed run-to-run variation.

Belebele. We also evaluated 27 of the 54 runs on Belebele (Bandarkar et al., 2024). Across these runs, aggregate accuracy ranges from 0.2259 to 0.2715, compared with a chance level of 0.25. Because performance remains approximately at chance, we do not use Belebele in our analyses.