# Parity-Aware Byte-Pair Encoding: Improving Cross-lingual Fairness in Tokenization

Anonymous ACL submission

## Abstract

Tokenization is the first—and often least scrutinized—step of most NLP pipelines. Standard algorithms for learning tokenizers rely on frequency-based objectives, which favor languages dominant in the training data and consequently leave lower-resource languages with tokenizations that are disproportionately longer, morphologically implausible, or even riddled with <UNK> placeholders. This phenomenon ultimately amplifies computational and financial inequalities between users from different language backgrounds. To remedy this, we introduce Parity-aware Byte Pair Encoding (BPE), a variant of the widely-used BPE algorithm. At every merge step, Parity-aware BPE maximizes the compression gain of the currently worst-compressed language, trading a small amount of global compression for cross-lingual parity. We find empirically that Parity-aware BPE leads to more equitable token counts across languages, with negligible impact on global compression rate and no substantial effect on LM performance in downstream tasks.

## 1 Introduction

At a time of rapid innovation and constant change in natural language processing (NLP), tokenization continues to be a foundational and comparatively stable component of NLP pipelines. Tokenization is the transformation of raw sequences of bytes[1] into sequences of byte-spans, *i.e.*, subwords; it enables computational efficiency and provides essential inductive biases by defining meaningful textual units. This design choice can have a major impact on various aspects of model performance (Bostrom and Durrett, 2020; Ali et al., 2024; Goldman et al., 2024).

The predominant tokenization algorithms—Byte Pair Encoding (BPE; Sennrich et al., 2016) and UnigramLM (Kudo, 2018), for example—select the vocabulary by maximizing frequency-based objectives computed over an entire training corpus. In multilingual corpora, this global criterion inevitably favors the languages with the greatest representation. Under vocabulary size constraints, subwords that primarily benefit high-resource languages are preferentially included, often at the expense of those needed for lower-resource languages. This bias has both qualitative and economic consequences. NLP models trained on fragmented or semantically incoherent tokenizations lose valuable inductive biases and tend to perform worse. At the same time, texts in lower-resource languages—often tokenized into more tokens—incur higher computational costs from language-model-based services charging based on token count, which disproportionately burdens users of underrepresented languages and exacerbates existing inequalities.

In the effort to mitigate these inequities, we introduce Parity-aware BPE. The classic version of BPE learns its vocabulary by repeatedly selecting the subword pair with the highest corpus-level co-occurrence count; it adds the concatenation of this pair to the vocabulary and replaces all pair co-occurrences with the new symbol.[2] Parity-aware BPE is a simple variant of this algorithm, retaining the iterative framework but redefining the merge selection rule: at each step, it computes co-occurrence statistics separately for each language and then uses statistics from the language with the current worst compression rate for selecting the next merge. In other words, instead of greedily maximizing a global objective, Parity-aware BPE performs a "fair-max" update that progressively equalizes string compression

[1] Early work considered characters the "base unit" of strings, but raw bytes have become popular because their fixed 256-symbol vocabulary can encode any character from any encoding, eliminating out-of-vocabulary issues.

[2] This process acts as a form of data compression, replacing frequent subword sequences with shorter representations.

rates across languages. Notably, this modification affects only vocabulary learning; the inference procedure remains the same as in Classical BPE.

Empirically, Parity-aware BPE leads to better token-count parity across languages than Classical BPE while maintaining comparable global compression. Across 13 multilingual benchmarks, models trained with a Parity-aware tokenizer match or exceed downstream performance compared to those trained with a Classical BPE tokenizer. Fairness metrics improve significantly, reflecting a more even distribution of token costs without sacrificing efficiency or downstream performance. In short, Parity-aware BPE reduces tokenizer-induced disparities between languages, enabling more equitable resource allocation and balanced performance across diverse linguistic settings.

## 2 Text Tokenization

Text can be decomposed at many granularities: graphemes, Unicode code points, or multi-character tokens; but at the most fundamental digital layer, every string is represented as a sequence of **bytes**, the foundation on which all other units are constructed. Let $b \in \mathcal{B} = \{0, \ldots, 255\}$ denote an individual byte. A finite **byte-string** is written $\mathbf{b} \in \mathcal{B}^*$, where $\mathbf{b} = b_1 b_2 \cdots b_{|\mathbf{b}|}$. Throughout, bytes are treated as atomic symbols. Note that all subsequent definitions apply if one substitutes bytes with another finite alphabet of atomic units.

### 2.1 Byte-level Tokenizers

In plain terms, **tokenization** is the act of mapping raw byte-strings (sequences of bytes) to sequences of subwords. A **tokenizer** specifies the rules that perform this mapping—as well as the rules that convert sequences of subwords back into byte-strings. The following components can define a tokenizer:

- **Vocabulary** $\mathcal{V} \subset \mathcal{B}^+$: a finite set of non-empty byte-spans, often called *subwords*.[3]
- **Tokenization function** $\tau : \mathcal{B}^* \to \mathcal{V}^*$: a mapping from byte-strings $\mathbf{b}$ to sequences of tokens $\mathbf{v} = v_1, v_2, \ldots$.
- **Detokenization function** $\perp : \mathcal{V}^* \to \mathcal{B}^*$: a mapping from sequences of subwords to byte-strings. This operation is often just simple string concatenation (denoted as $\circ$) of subwords' corresponding byte spans: $\perp(v_1, \ldots, v_n) = v_1 \circ v_2 \circ \cdots \circ v_n$

[3]To guarantee representability of any byte-string, we assume $\mathcal{V}$ includes all singleton bytes: $\mathcal{B} \subseteq \mathcal{V}$.

**Pre-tokenization and Normalization.** Many tokenization algorithms include a *pre-tokenization* (and often a normalization) step that segments or rewrites raw byte strings according to deterministic criteria. Pre-tokenization can encompass several operations, including Unicode normalization or splitting on whitespace. Notably, pre-tokenization determines subword boundaries, and thus also determines the set of possible candidates for the vocabulary as well as the attainable compression rate. For example, if whitespace is used as a subword boundary, languages without explicit whitespace (*e.g.*, Chinese, Japanese) or with rich morphology may have the potential for higher compression. For simplicity, we assume this step is baked into $\tau$.

### 2.2 Text Compression

*Why does mapping a raw byte-string to a sequence of larger subword tokens help a language model (LM)?* The precise inductive biases this process imbues remain an open research question (Zouhar et al., 2023a; Schmidt et al., 2024), but one plausible explanation is the *compression* it provides: a good tokenizer tends to map each input $\mathbf{b}$ to a shorter sequence of tokens, reducing the length of the model's effective input and, potentially making learning easier. At the very least, it can significantly reduce model-side computations.

For a fixed tokenizer $(\mathcal{V}, \tau, \perp)$ we define the **compression rate** of a byte-string $\mathbf{b}$ as

$$\mathrm{CR}(\mathbf{b}; \tau) \stackrel{\text{def}}{=} \frac{|\mathbf{b}|_u}{|\tau(\mathbf{b})|} \tag{1}$$

where $|\mathbf{b}|_u$ denotes the length of $\mathbf{b}$ in terms of a given **normalization unit** $u$ (*e.g.*, characters, words, lines, or simply bytes). In words, $\mathrm{CR}(\mathbf{b}; \tau)$ measures the multiplicative factor by which our original sequence length is reduced after tokenization. A higher CR indicates stronger compression.

We are generally interested in a tokenizer's average compression, which can be estimated from a corpus $\mathcal{D}$:

$$\mathrm{CR}(\mathcal{D}; \tau) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}|} \sum_{\mathbf{b} \in \mathcal{D}} \frac{|\mathbf{b}|_u}{|\tau(\mathbf{b})|} \tag{2}$$

This quantity provides an estimate of the average number of tokens that an autoregressive LM must process per unit $u$ of raw text. Tokenizers differ in how small they can make $\mathrm{CR}(\mathcal{D}; \tau)$ while remaining lossless. Further, this rate can vary across strings from different languages, which

motivates our last definition: language-specific compression rate.

Let $\mathcal{L} = \{\ell^{(r)}\}_{r=1}^{R}$ be our set of languages and $\mathcal{M} = \{(\mathbf{b}^{(s)}, \ell^{(s)})\}_{s=1}^{S}$ be a labeled multilingual corpus, *i.e.*, a corpus where each byte-string $\mathbf{b}^{(s)}$ is labeled with its respective language. For a fixed tokenizer, we define a language's compression rate as

$$\mathrm{CR}(\ell; \tau) \stackrel{\text{def}}{=} \mathrm{CR}(\mathcal{D}_\ell; \tau) \tag{3}$$

where $\mathcal{D}_\ell = \{\mathbf{b}^{(s)} : (\mathbf{b}^{(s)}, \ell^{(s)}) \in \mathcal{M},\ \ell^{(s)} = \ell\}$.

**Compression Rate as a Notion of Fairness.** Many commercial technology services offer APIs that bill per token; services' processing speeds also scale with the number of tokens in the input and output. Token counts thus dictate both the economics and the latency of these services. For a given byte sequence $\mathbf{b}$, the number of tokens produced by tokenization function $\tau$ is determined by the tokenizer's compression rate $\mathrm{CR}(\mathbf{b}; \tau)$. $\mathrm{CR}(\ell; \tau)$—the expected compression rate over sequences from language $\ell$—is thus a direct proxy for the average per-language cost (and expected latency) of using one of these services. Variance in $\mathrm{CR}(\ell; \tau)$ across languages thus implies different user costs purely as a function of language choice (Petrov et al., 2023), and whether $\mathrm{CR}(\ell; \tau)$ is comparable across languages is therefore one way of measuring tokenizer fairness. Byte Pair Encoding, discussed next, optimizes for compression rate across a corpus without regard for language. Our subsequent adjustment to Classical BPE adds an auxiliary objective of equalizing $\mathrm{CR}(\ell; \tau)$ across languages.

### 2.3 Byte Pair Encoding

*Here we provide a detailed description of the classic version of BPE; readers already familiar with the algorithm can skip this subsection.*

Byte Pair Encoding (BPE; Sennrich et al., 2016) is one popular algorithm for creating a tokenizer adapted from the byte-pair compression scheme of Gage (1994). In short, BPE tokenizes text by iteratively *merging* adjacent tokens whose token-types (*i.e.*, subwords) were observed to co-occur frequently in the training data.

The notion of a **merge** lets us formalize this procedure. A merge is defined as an ordered pair $m = (v, v')$ with $v, v' \in \mathcal{V}$. The application of a merge to a token sequence replaces every bigram token $v, v'$ by a single token $v \circ v'$. Each bigram token replacement shortens the token sequence by exactly one token, thereby *compressing* the sequence. To tokenize a piece of text with a BPE tokenizer, we start from its representation as a byte-string, *i.e.*, a sequence of base bytes—all of which necessarily appear in our tokenizer's vocabulary. We then iteratively apply a given list of merges to that sequence. Note that because the merge list is fixed in advance, the encoding is deterministic. Intuition for the merge procedure is perhaps best acquired by a small example:

**Example 2.1** (Example of the iterative application of merge sequence $\mathbf{m}$ to byte sequence $\mathbf{b}$ **).**

$$\begin{aligned} \mathbf{m} &= [(\mathsf{b}, \mathsf{a}), (\mathsf{ba}, \mathsf{b})]; \quad \mathbf{b} = \mathsf{babab} \\ \mathbf{v}_0 &= \mathsf{b}, \mathsf{a}, \mathsf{b}, \mathsf{a}, \mathsf{b} \\ \text{Step 1: } \mathsf{b}, \mathsf{a} \to \mathsf{ba} &\implies \mathbf{v}_1 = \mathsf{ba}, \mathsf{ba}, \mathsf{b} \\ \text{Step 2: } \mathsf{ba}, \mathsf{b} \to \mathsf{bab} &\implies \mathbf{v}_2 = \mathsf{ba}, \mathsf{bab} \end{aligned}$$

In terms of our earlier tokenizer notation, a BPE tokenizer is defined as follows:

- $\mathcal{V} = \mathcal{B} \cup \{v \circ v' : (v, v') \in \mathbf{m}\}$
- $\tau_{\mathbf{m}}$ carries out the procedure described above, *i.e.*, it applies each $m_k$ to an input byte-string $\mathbf{b}$ in the prescribed order. In example 2.1, $\tau_{\mathbf{m}}(\mathbf{b}) = \mathsf{ba}, \mathsf{bab}$.
- $\perp(v_1, \ldots, v_n) = v_1 \circ v_2 \circ \cdots \circ v_n$

We use the $\mathbf{m}$ subscript here to make explicit the tokenization function's dependence on $\mathbf{m}$.

**Learning $\mathbf{m}$.** The BPE algorithm seeks the merge list $\mathbf{m}^*$ (subject to a size constraint $K$) that maximizes the compression rate of the given corpus:

$$\mathbf{m}^* = \max_{\mathbf{m}:|\mathbf{m}|=K} \mathrm{CR}(\mathcal{D}; \tau_{\mathbf{m}}) \tag{4}$$

BPE takes a greedy approach to choosing $\mathbf{m}$, finding an approximate solution to eq. 4 (Zouhar et al., 2023b). It starts with the singleton-byte vocabulary $\mathcal{V}_0 = \mathcal{B}$ and repeatedly greedily enlarges the vocabulary. At each of $K$ steps, the current tokenizer $\tau_{\mathbf{m}_{<k}}$ is applied to the entire training corpus, and the algorithm counts how often every adjacent pair of tokens occurs. The subword-type pair with the highest count, which we denote as $(v^\star, v'^\star)$, is deemed the most "compressive." Its concatenation $v^\star \circ v'^\star$ is added to the vocabulary, the merge $m_k = (v^\star, v'^\star)$ is recorded, and every occurrence of the bigram $(v^\star, v'^\star)$ in the corpus is replaced by the new token so the next iteration works with updated token sequences. Repeating this process $K$ times yields the ordered list $\mathbf{m} = [m_1, \ldots, m_K]$ and the final vocabulary $\mathcal{V}_K$. When encoding a new text, $\tau_{\mathbf{m}}$ simply applies these merges in the same order. The pseudocode for the algorithm is provided in Alg. 1 in §A.

## 3 Parity-aware Byte Pair Encoding

Classical BPE chooses merges that maximize a *global* frequency objective, implicitly favoring the compression of languages with a larger presence in the training corpus. Here we introduce **Parity-aware BPE**, which replaces this global objective with a *max–min* criterion: at every step, it selects the merge that most improves the language currently suffering the poorest compression rate. In this section, we formalize the objective and describe the resulting algorithm.

### 3.1 Greedy min-max objective

Our adjustment to the Classical BPE objective (eq. 4) explicitly encodes our earlier notion of tokenizer fairness: equality across per-language compression rates. Formally, parity-aware BPE seeks a merge list $\mathbf{m} = [m_1, \ldots, m_K]$ that maximizes the *minimum* compression rate across languages:

$$\mathbf{m}^\star = \max_{\mathbf{m}:|\mathbf{m}|=K} \min_{\ell} \mathrm{CR}(\ell; \tau_{\mathbf{m}}) \tag{5}$$

This min-max objective trades a small amount of global compression for fairness across languages.

### 3.2 Algorithm

Parity-aware BPE retains the greedy iterative framework of Classical BPE but changes *which* statistics are inspected each time a merge is added. At merge step $k$, it identifies the language with the worst compression under the tokenizer defined by the merge list thus far ($\mathbf{m}_{<k}$)

$$\ell^\star \;=\; \arg\min_{\ell \in \mathcal{L}} \mathrm{CR}(\ell; \tau_{\mathbf{m}_{<k}}) \tag{6}$$

To choose the next merge, it uses the same maximum pair count criterion as Classical BPE, albeit with pair counts computed over only $\mathcal{D}_{\ell^\star}$—the portion of the corpus corresponding to $\ell^\star$. The rest of the algorithm follows the Classical BPE procedure: the chosen merge is applied to all texts (*i.e.*, across $\mathcal{D}_\ell\ \forall \ell$)[4] and the procedure is repeated for $k = 1, \ldots, K$, yielding the final merge list $\mathbf{m}$. We provide pseudocode in Alg. 2.

**Cross-lingual Compression Rate Comparison.** Parity-aware BPE relies on the comparison of $\mathrm{CR}(\ell; \tau_{\mathbf{m}})$ across different $\ell$. The choice of normalization unit $u$ has a large impact on the measured $\mathrm{CR}(\ell; \tau_{\mathbf{m}})$ and even when $u$ is held constant across measurements for different languages, if not considered carefully, the choice can introduce bias into the comparison. As concrete examples, certain normalization units are more appropriate in some languages than in others, *e.g.*, whitespace-delimited "words" are ill-defined in many languages; although principled and universal, even normalizing by number of bytes can skew perceived compression because scripts differ greatly in average number of bytes per character (*e.g.*, ASCII vs. UTF-8 CJK). Parallel corpora provide a principled solution: computing compression rates over aligned texts (sentences, lines, or documents) normalizes by content, making cross-language comparisons more meaningful. We therefore recommend the use of a parallel corpus for computing eq. 6. Notably, this evaluation corpus need not be the same one used for computing subword pair frequency statistics, for which a larger corpus with only language annotation is necessary. For generality, we thus differentiate between the corpora used to compute frequency statistics and in computing eq. 6, referring to them as our training and development datasets, respectively. Alg. 2 makes this difference explicit. We present experimental results both with a separate, parallel development set and using a single (not parallel) multilingual dataset for all computations.

**Complexity and Data Requirements.** Relative to Classical BPE, Parity-aware BPE incurs only a $O(|\mathcal{L}|)$ overhead per-merge from recomputing the language-specific compression rates on the dev set. Parity-aware BPE retains the same asymptotic complexity as Classical BPE, requiring only some modest additional bookkeeping. The need for a parallel multilingual corpus can at first seem prohibitive, but several pragmatic design choices can reduce the burden of this requirement. A small, aligned dataset suffices to drive the max-min decision in eq. 6, and the training dataset need not have this level of annotation. In addition, automatic language ID tools or script heuristics can help provide language labels when none are readily available. Also note that only the BPE learning phase differs; there is no algorithmic change to the tokenization function itself.

### 3.3 Algorithmic Variants

Preliminary experiments have shown several challenges with parity-aware BPE, which we address by introducing two variants.

[4] Crucially, this is what distinguishes our algorithm from a combination of monolingual merge lists (Petrov et al., 2023), allowing us to find more "compressive" merges.

**Hybrid parity-aware BPE.** Model developers may want to include and tokenize data for which parallel data is not available or where this concept does not even apply, such as programming code. Also, they may not want to guarantee full parity, but still give a high weight to global compression. We support these goals with a hybrid learning algorithm that uses the global objective of Classical BPE (eq. 4) for the first $K$ merges, then switches to the parity-aware objective (eq. 5) for another $J$ merges. $K$ and $J$ can be chosen by model developers to trade off global compression and fairness according to their priorities.

**Moving-window balancing.** There may be a point where the compression in a language no longer or barely improves, even if it is repeatedly chosen for the next merge. This could happen if the development dataset (the dataset used to choose the language) is too small or does not match the domain or language variant of the training dataset,[5] or if $|\tau(\mathbf{b})|$ approaches the length of the pre-tokenized sequence. To prevent our algorithm from being "stuck" selecting the same language exclusively, we track the $W$ most recent languages selected in eq. 6, and do not select a language if it occurs more than $\alpha\frac{W}{|\mathcal{L}|}$ times in this moving window.

## 4 Experimental Setup

We conduct experiments to evaluate the effectiveness of Parity-aware BPE, comparing it against baseline tokenization methods: Classical BPE and UnigramLM. All tokenizers are byte-level.

### 4.1 Tokenizer Training

**Training Data.** We train tokenizers using the multilingual C4 (mC4) corpus (Xue et al., 2021; Raffel et al., 2020).[6] For choosing the focus language at each merge step when training Parity-aware BPE tokenizers (*i.e.*, computing eq. 6), we use the dev portion of FLORES+ (NLLB Team et al., 2024) as our multilingual development corpus—except for the *no-dev* systems, for which the training corpus is used to measure compression rate with bytes as normalization unit. To investigate how the number of languages, their linguistic diversity, and the variety of writing systems influence tokenizers, we consider two language sets: one with 30 languages (*30-lang*) and another with 60 languages (*60-lang*). For each of these language sets, we create two dataset versions: one with uniform quantities of data per language (*balanced*) and one with per-language quantities proportional to amounts in the mC4 dataset (*unbalanced*). We present results for the *unbalanced* datasets here, as this is arguably the more realistic setting, with results for the *balanced* setting shown in §D. To enable tokenizer analyses as a function of different dataset qualities, we categorize the languages in each set based on the amount of training data available and the script family, performing some of our analyses by these categories. Languages with $>$ 1M examples are considered high-resource; those with 500k $-$ 1M examples are medium-resource; and those with fewer than $<$ 500k examples are classified as low-resource. The full list of languages included in each set and the script family groupings are presented in Table 8.

**Hyperparameter Settings.** We look at vocabulary sizes $128k$ and $256k$. For *hybrid* systems, we learn half of the merges using the global strategy, and the second half using the parity-aware strategy. For systems with moving-window balancing (*window*), we use a window size of 100, and $\alpha = 2$. More information is provided in Appendix §C.

### 4.2 Evaluation

Our evaluations consist of task-independent tokenizer properties (intrinsic metrics) and downstream model performance (extrinsic metrics).

#### 4.2.1 Intrinsic Metrics

We measure a variety of intrinsic tokenizer metrics on the devtest portion of FLORES+. All metrics are computed both globally and per-language to capture language-specific tokenization behavior. Normalization units differ across metrics, both in the effort to control for confounding factors and to tailor the metric to the tokenizer quality it is trying to measure. For example, morphologically motivated units can better reflect linguistic structure, and using character or document–level units can partially normalize the large differences in average bytes-per-character observed across writing systems (*e.g.*, Latin vs. UTF-8–encoded CJK[7] scripts). For the sake of space, we provide brief metric descriptions here; more detailed definitions and formulae for all metrics can be found in §B.

[5] Kreutzer et al. (2022) discuss possible quality issues such as wrong or ambiguous language codes.

[6] https://huggingface.co/datasets/allenai/c4

[7] Chinese, Japanese and Korean scripts.

- **Fertility** measures the average number of tokens produced per normalization unit by a tokenizer; whitespace-delimited words are often the unit of interest (and are the units used in our computations). In this case, fertility quantifies how many tokens (on average) a word is broken up into.
- **Compression Rate (CR)** (as defined in §2) is a measure of the degree to which a unit of text has been shrunk after applying the given tokenizer (higher is better). We use documents as the normalization unit.
- **Vocabulary Utilization** is the fraction of the tokenizer's vocabulary that actually appears in the evaluation corpus. Low utilization for a language signals wasted capacity or—when there are large differences across languages—biased vocabulary allocation.
- **Tokenizer Fairness Gini** (Meister, 2025) adapts the Gini coefficient to the per-language tokenization cost distribution (*e.g.*, tokens per line (document) in a parallel corpus). Values near 0 mean equal cost across languages; values closer to 1 indicate inequality.
- **MorphScore** (Arnett et al., 2025) measures how well token boundaries align with true morpheme boundaries, computed as morpheme-level precision/recall (and F1). High scores mean tokens respect morphological structure; low precision implies over-segmentation, while low recall may suggest under-segmentation.

For completeness, we also track Type–Token Ratio and Average Token Rank (vocabulary diversity; Limisiewicz et al., 2023) as well as Rényi entropies (distributional concentration; Zouhar et al., 2023a), evaluated using the TokEval suite (Meister, 2025).

#### 4.2.2 Extrinsic Metrics.

For extrinsic evaluation, we train models using different tokenizers and assess their performance across a range of downstream tasks.

**Model Architecture and Pretraining Data.** We train decoder-only Transformer models (Vaswani et al., 2017) following the LLaMA architecture (Touvron et al., 2023) with 3 billion (3B) parameters. Full details on model configurations and training parameters are provided in §F. Models are trained on the FineWeb2 corpus (Penedo et al., 2025). We adopt temperature sampling with $\tau = 3.3$, following recommendations from prior work (Raffel et al., 2020; Conneau et al., 2020). We use the total 100B tokens to train each model.

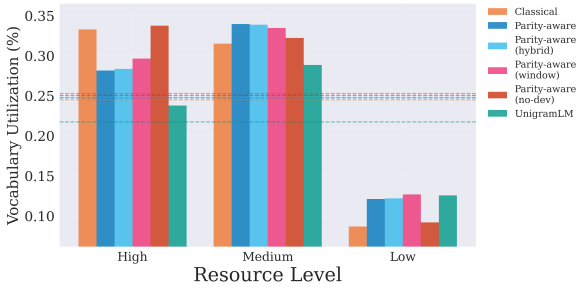


Figure 1: Vocabulary utilization for $128k$ tokenizers on the *unbalanced 30-lang* grouped by dataset resource levels. −− lines indicate macro-averages across groups.

**Benchmarks.** We evaluate the models using perplexity on a held-out validation set from the respective pretraining datasets. In addition, we assess downstream performance on a suite of multilingual benchmarks. Results are aggregated per language to produce a score for each model-language pair. A full list of benchmarks and aggregation procedures is provided in the §G.

## 5 Results and Analysis

We present and analyze results on both intrinsic and extrinsic metrics. Within a set of comparisons, we fix the data distribution (*balanced* or *unbalanced*) and vocabulary size ($128k$ or $256k$).

### 5.1 Intrinsic Evaluation

Results in Table 1 show that all variants of Parity-aware BPE outperform Classical BPE in terms of the Gini coefficient, indicating more equitable token costs per document across languages. Among these variants, the base Parity-aware BPE emerges as the "fairest" tokenizer. Notably, Classical BPE and the Parity-aware BPE variants attain almost identical compression and Rényi entropies; we take this as evidence that the parity-aware variants match global efficiency while redistributing it more evenly. In addition, Parity-aware variants reduce fertility and increase MorphScore and vocabulary utilization, signaling better alignment with morphological boundaries and fairer vocabulary allocation. As could perhaps be expected, the no-dev variant of Parity-aware BPE performs most similarly to Classical BPE, closing only part of the Gini gap but matching it almost exactly on every other metric. This observation demonstrates the importance of thoughtfully consideration of normalization units for the effectiveness of the algorithm; future work could address this by explicitly compensating for differences in cross-language length statistic, *e.g.*, by introducing a per-language

| Tokenizer | Type-Token Ratio | Fertility | Compression Rate | Rényi Entropy ($\alpha$=2.5) | Gini Coefficient | MorphScore Precision | MorphScore Recall |
|---|---|---|---|---|---|---|---|
| Classical | 0.0743 | 4.260 ± 0.049 | 0.0303 ± 0.0001 | **8.13** | 0.064 | 0.412 ± 0.051 | 0.456 ± 0.049 |
| UnigramLM | 0.0475 | 4.612 ± 0.042 | 0.0228 ± 0.0001 | 4.68 | 0.094 | 0.153 ± 0.037 | 0.268 ± 0.053 |
| Parity-aware | 0.0765 | 4.204 ± 0.049 | 0.0300 ± 0.0001 | 8.12 | **0.011** | 0.407 ± 0.051 | 0.457 ± 0.049 |
| Parity-aware (hybrid) | 0.0770 | **4.191 ± 0.049** | 0.0303 ± 0.0001 | 8.10 | 0.018 | 0.412 ± 0.051 | 0.457 ± 0.049 |
| Parity-aware (window) | 0.0788 | 4.219 ± 0.050 | 0.0302 ± 0.0001 | 8.11 | 0.013 | 0.405 ± 0.049 | 0.453 ± 0.047 |
| Parity-aware (window+hybrid) | **0.0794** | 4.203 ± 0.050 | **0.0305 ± 0.0001** | 8.09 | 0.022 | 0.416 ± 0.049 | 0.460 ± 0.047 |
| Parity-aware (no-dev) | 0.0772 | 4.310 ± 0.050 | 0.0303 ± 0.0001 | 8.12 | 0.059 | **0.423 ± 0.051** | **0.466 ± 0.049** |

Table 1: Intrinsic evaluation of $128k$ tokenizers on the (*unbalanced*) *30-lang* dataset. Values are global statistics across the parallel corpus, except for MorphScore, which is macro-averaged across available languages.

| Language | Classical BPE | Parity-aware (hybrid) | Parity-aware (window+hybrid) | Random |
|---|---|---|---|---|
| Arabic | 38.19 ± 2.90 | **39.04** ± 2.89 | 38.84 ± 2.90 | 32.00 |
| Bengali | 24.95 ± 3.09 | 23.54 ± 2.98 | 23.91 ± 3.01 | **25.00** |
| German | 32.92 ± 3.14 | 34.78 ± 3.66 | **36.82** ± 4.04 | 30.62 |
| Greek | 41.95 ± 3.18 | 42.55 ± 3.22 | **43.16** ± 3.25 | 37.50 |
| Spanish | 37.53 ± 2.66 | 38.83 ± 2.71 | **39.30** ± 2.75 | 32.77 |
| Persian | **42.80** ± 5.39 | 39.15 ± 5.27 | 39.15 ± 5.27 | 25.00 |
| French | **38.67** ± 3.90 | 36.59 ± 2.84 | 37.10 ± 2.82 | 32.00 |
| Hindi | **33.92** ± 2.25 | 33.92 ± 2.24 | 33.86 ± 2.24 | 30.62 |
| Indonesian | 38.95 ± 2.62 | **40.55** ± 2.66 | 40.46 ± 2.66 | 35.00 |
| Italian | 32.82 ± 2.86 | **35.01** ± 3.00 | 34.62 ± 2.98 | 27.22 |
| Japanese | 37.43 ± 2.39 | **37.45** ± 2.39 | 37.43 ± 2.39 | 34.00 |
| Korean | 33.00 ± 5.22 | 33.00 ± 5.22 | **34.33** ± 5.29 | 25.00 |
| Polish | 29.75 ± 2.50 | **31.14** ± 2.60 | 28.97 ± 2.49 | 23.75 |
| Portuguese | **33.63** ± 2.81 | 33.15 ± 2.77 | 33.06 ± 2.77 | 27.50 |
| Russian | 36.36 ± 2.27 | 36.21 ± 2.26 | **36.57** ± 2.28 | 32.77 |
| Tamil | 31.32 ± 2.81 | **32.25** ± 2.90 | 32.19 ± 2.90 | 31.25 |
| Telugu | 32.73 ± 2.61 | **33.52** ± 2.61 | 33.26 ± 2.61 | 30.00 |
| Turkish | **39.04** ± 2.89 | 38.46 ± 2.83 | 37.89 ± 2.75 | 35.00 |
| Vietnamese | 33.69 ± 2.31 | **33.87** ± 2.27 | 33.80 ± 2.30 | 29.50 |
| Chinese | 38.43 ± 2.11 | **38.58** ± 2.11 | 38.32 ± 2.10 | 35.00 |
| English | 43.04 ± 1.84 | **44.15** ± 1.85 | 43.74 ± 1.85 | 35.50 |
| Thai | 40.76 ± 1.62 | 40.96 ± 1.63 | **41.06** ± 1.63 | 37.50 |

Table 2: Average downstream performance (accuracy %) across 13 multilingual benchmarks (tokenizers trained on the (*unbalanced*) *30-lang* dataset with $128k$ vocab size). The **Random** column shows the expected accuracy of a random classifier. Best performance per language is bolded. Benchmark details for each language are provided in §G and Table 9, respectively.

multiplicative factor.[8] By contrast, the hybrid and window variants land between Parity-aware and Classical BPE on many metrics: they recover a small slice of global compression while reducing the Gini inequity coefficient of Classical BPE by roughly three-quarters and achieving the lowest fertility of all runs. Taken together, the variants outline a smooth fairness–efficiency frontier, allowing practitioners to select the point that best suits their resource constraints and fairness targets.

Fig. 1 presents vocabulary utilization grouped by resource tier, as defined in §4.1. For high-resource languages, the results indicate that parity-aware BPE (no-dev) performs comparably to Classical BPE in terms of vocabulary utilization, while the other variants provide worse vocabulary utilization. In contrast, for low- and medium-resource languages, the hybrid and window variants achieve higher vocabulary utilization, highlighting their effectiveness at providing fairer vocabulary allocation across languages. We show per-language results for compression rate and vocabulary utilization in Fig. 2 and 3. We observe that Parity-aware tokenizers attain substantially more uniform compression across languages. While they also generally achieve higher vocabulary utilization rates across most languages, the level of benefit varies by language.

**Vocabulary Size.** Repeating the experiment with a $256k$ vocabulary (Table 7 and Fig. 4) yields the same conclusion: parity-aware BPE tokenizers consistently outperform Classical BPE on cross-lingual fairness metrics, with minimal impact on performance-oriented measures.

### 5.2 Extrinsic Evaluation

Table 2 presents the performance of LMs trained with three different $128k$ tokenizers: Classical BPE, Hybrid Parity-aware BPE, and Hybrid Parity-aware BPE (moving-window), evaluated on the *30-lang* set. For each language, we report mean performance, standard errors, and a random baseline to account for varying benchmark counts (Table 9). The evaluation assesses Parity-aware BPE's impact on downstream performance, particularly in languages where Classical BPE is efficient. Results indicate that Parity-aware BPE maintains performance across languages: accuracy changes relative to Classical BPE are small. Models trained with the hybrid variant show a median per-language change in accuracy of $+0.19$ percentage points, with 14 languages improving and 6 declining; the window+hybrid leads to very similar changes in accuracy. These results confirm that parity-aware tokenizers can handle diverse languages without compromising LM performance. We show per-language perplexity results in Fig. 6 in §D. Here, we see that models trained with parity-aware tokenizers show much more uniform perplexity across languages, whereas Classical BPE yields a handful of languages with markedly higher perplexity.

[8] For example, without a multi-parallel corpus, developers can estimate the desired compression rate from a number of parallel corpora.

## 6 Related Work

**Multilingual Tokenization.** Despite their popularity, BPE and related subword tokenization methods often underperform in multilingual settings due to limited handling of spelling variation and morphological complexity (Bostrom and Durrett, 2020). A tokenizer's tokenization parity and fertility directly affect computational cost and model performance. Prior work has explored vocabulary allocation strategies: Zhang et al. (2022) show that larger vocabularies improve NMT robustness across scripts, while Gowda and May (2020) demonstrate that tuning BPE merges can mitigate sequence length issues. Rust et al. (2021) find that integrating specialized monolingual tokenizers into multilingual systems can boost performance; however, recent results indicate that optimal vocabulary size depends on the task and model (Dagan et al., 2024). For multilingual vocabulary construction, Chung et al. (2020) explore clustering-based sharing of subword units across languages, and Limisiewicz et al. (2023) propose an explicit tokenizer-merging algorithm to combine per-language vocabularies. Finally, tokenization-free models such as CANINE (Clark et al., 2022) and ByT5 (Xue et al., 2022) offer an alternative approach for improved multilingual handling.

**Tokenization Bias and Recent Advances.** Recent research highlights biases from tokenization in LLMs. While Wan (2022) argues that character- and byte-level representations are intrinsically fair,[9] other studies (Petrov et al., 2023; Ahia et al., 2023) show that tokenization differences across languages; even at character and byte levels; affect costs, latency, and contextual understanding. This has spurred efforts like Aya (Aryabumi et al., 2024) and methods to mitigate tokenization unfairness (Fujii et al., 2024; Abboud and Oz, 2024; Limisiewicz et al., 2024). Although newer character- and byte-level models use compression techniques such as entropy-based patching (Pagnoni et al., 2025), cross-lingual parity of these representations remains unstudied.

## 7 Discussion and Conclusion

Tokenizers optimized using standard algorithms and data can lead to disparities in users' costs and experiences as a result of their choice of language. Parity-optimized tokenization can remedy this by explicitly balancing compression across languages, enabling fairer treatment of users of low-resource languages. Parity-aware BPE implements this idea: it was designed to improve cross-lingual tokenization parity, and our experiments confirm that it does. On the unbalanced 30-language set, the Gini coefficient of per-line token costs falls from $0.064$ with Classical BPE to $0.011$ with our parity-aware variant while compression ratios for most variants stay competitive with Classical BPE, often improving when looking at averages across languages on the whole. Crucially, this fairness gain does not come at the expense of downstream quality: across 13 multilingual benchmarks, models trained with parity-aware tokenizers either outperform or stay within a single standard error of the Classical BPE baseline for every tested language (Table 2).

Overall, the trade-offs required for using parity-aware BPE are minimal. From the model-developer's perspective, parity-aware BPE is a drop-in replacement: it requires no architectural changes and minimal changes to the tokenizer pipeline. Concretely, inference remains exactly the same as in Classical BPE. During the learning stage, the algorithm adds only an $\mathcal{O}(\mathcal{L})$ pass per merge for recomputing language-level compression rates on a dev corpus, leaving the asymptotic complexity identical to Classical BPE. Further, we observe empirically that a small, sentence-aligned development set is sufficient to drive the "fair-max" decision (§C.3). When resource or domain mismatches make full equality undesirable, hybrid and moving-window variants further let practitioners trade off global compression versus strict parity; our empirical results validate that these variants perform well in practice.

Making the tokenization step of the NLP pipeline more equitable is therefore not just desirable but feasible. Parity-aware BPE offers a clear avenue towards this goal by building fairness into the tokenizer itself. With a simple modification—selecting merges that benefit the worst-compressed language—it substantially reduces the hidden "token tax" imposed on speakers of low-resource languages without sacrificing compression or task accuracy. Future work can push this agenda further by extending parity objectives to alternative tokenization schemes and other modalities such as speech and vision, as well as by developing benchmarks and metrics for fairness assessments in tokenization beyond compression parity.

[9] They report more random performance with these representations, which we do not view as "fairer."

## Limitations

Our study uses parallel corpora to estimate per-language costs; in domains where aligned documents are unavailable or difficult to obtain, using unaligned corpora and alternative normalization units for making the language choice may introduce bias. While we consider 60 languages and two vocabulary sizes, the interplay between tokenization parity and model scaling still needs to be explored for much larger models and for code or multimodal inputs. Finally, fairness here is defined purely in terms of token counts. While we measure other potential quantification of fairness (*e.g.*, morphological alignment), there are still other notions that are unaccounted for. We leave optimization for these metrics during tokenizer learning to future work.

## References


Khadige Abboud and Gokmen Oz. 2024. Towards equitable natural language understanding systems for dialectal cohorts: Debiasing training data. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 16487–16499. ELRA and ICCL.

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David R. Mortensen, Noah A. Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9904–9923. Association for Computational Linguistics.

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. 2024. Tokenizer choice for LLM training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico. Association for Computational Linguistics.

Catherine Arnett, Marisa Hudspeth, and Brendan O'Connor. 2025. Evaluating Morphological Alignment of Tokenizers in 70 Languages. In *Proceedings of the ICML 2025 Tokenization Workshop (TokShop)*.

Viraat Aryabumi, John Dang, Dwarak Talupuru, Saurabh Dash, David Cairuz, Hangyu Lin, Bharat Venkitesh, Madeline Smith, Kelly Marchisio, Sebastian Ruder, et al. 2024. Aya 23: Open weight releases to further multilingual progress. *CoRR*, abs/2405.15032.

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

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Michael Chen, Mike D'Arcy, Alisa Liu, Jared Fernandez, and Doug Downey. 2019. CODAH: An adversarially-authored question answering dataset for common sense. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 63–69. Association for Computational Linguistics.

Hyung Won Chung, Dan Garrette, Kiat Chuan Tan, and Jason Riesa. 2020. Improving multilingual models with language-clustered vocabularies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4536–4546. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. CANINE: Pre-training an efficient tokenization-free encoder for language representation. *Trans. Assoc. Comput. Linguistics*, 10:73–91.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451. Association for Computational Linguistics.

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

Gautier Dagan, Gabriel Synnaeve, and Baptiste Rozière. 2024. Getting the most out of your tokenizer for pre-training and domain adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Hiroki Iida, Masanari Ohi, Kakeru Hattori, Hirai Shota, Sakae Mizuki, Rio Yokota, and Naoaki Okazaki. 2024. Continual pre-training for cross-lingual LLM adaptation: Enhancing japanese language capabilities. In *First Conference on Language Modeling*.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

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

Thamme Gowda and Jonathan May. 2020. Finding the optimal vocabulary size for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP*, volume EMNLP 2020 of *Findings of ACL*, pages 3955–3964. Association for Computational Linguistics.

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. 2023. Lighteval: A lightweight framework for llm evaluation.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben allal, Leandro Von Werra, and Martin Jaggi. 2024. Scaling laws and compute-optimal training beyond fixed training durations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Momchil Hardalov, Todor Mihaylov, Dimitrina Zlatkova, Yoan Dinkov, Ivan Koychev, and Preslav Nakov. 2020. EXAMS: A multi-subject high school examinations dataset for cross-lingual and multilingual question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5427–5444, Online. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 66–75. Association for Computational Linguistics.

Viet Lai, Chien Nguyen, Nghia Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan Rossi, and Thien Nguyen. 2023. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 318–327. Association for Computational Linguistics.

Tomasz Limisiewicz, Jiří Balhar, and David Mareček. 2023. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. In *Findings of the Association for Computational Linguistics*, pages 5661–5681. Association for Computational Linguistics.

Tomasz Limisiewicz, Terra Blevins, Hila Gonen, Orevaoghene Ahia, and Luke Zettlemoyer. 2024. MYTE: Morphology-driven byte encoding for better and fairer multilingual language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 15059–15076. Association for Computational Linguistics.

Bill Yuchen Lin, Seyeon Lee, Xiaoyang Qiao, and Xiang Ren. 2021. Common sense beyond English: Evaluating and improving multilingual language models for commonsense reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1274–1287, Online. Association for Computational Linguistics.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022a. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona T. Diab, Veselin Stoyanov, and Xian Li. 2022b. Few-shot learning with multilingual generative language models. In

*Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 9019–9052. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Clara Meister. 2025. TokEval: A tokenizer analysis suite.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2023. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15991–16111, Toronto, Canada. Association for Computational Linguistics.

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. 2024. Scaling neural machine translation to 200 languages. *Nature*, 630(8018):841–846.

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

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. 2025. FineWeb2: One pipeline to scale them all – adapting pretraining data processing to every language. *CoRR*, abs/2506.20920.

Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Angelika Romanou, Negar Foroutan, Anna Sotnikova, Sree Harsha Nelaturu, Shivalika Singh, Rishabh Maheshwary, Micol Altomare, Zeming Chen, Mohamed A. Haggag, Snegha A, Alfonso Amayuelas, Azril Hafizi Amirudin, Danylo Boiko, Michael Chang, Jenny Chim, Gal Cohen, Aditya Kumar Dalmia, Abraham Diress, Sharad Duwal, Daniil Dzenhaliou, Daniel Fernando Erazo Florez, Fabian Farestam, Joseph Marvin Imperial, Shayekh Bin Islam, Perttu Isotalo, Maral Jabbarishiviari, Börje F. Karlsson, Eldar Khalilov, Christopher Klamm, Fajri Koto, Dominik Krzemiński, Gabriel Adriano de Melo, Syrielle Montariol, Yiyang Nan, Joel Niklaus, Jekaterina Novikova, Johan Samir Obando Ceron, Debjit Paul, Esther Ploeger, Jebish Purbey, Swati Rajwal, Selvan Sunitha Ravi, Sara Rydell, Roshan Santhosh, Drishti Sharma, Marjana Prifti Skenduli, Arshia Soltani Moakhar, Bardia Soltani Moakhar, Ayush Kumar Tarun, Azmine Toushik Wasi, Thenuka Ovin Weerasinghe, Serhan Yilmaz, Mike Zhang, Imanol Schlag, Marzieh Fadaee, Sara Hooker, and Antoine Bosselut. 2025. INCLUDE: Evaluating multilingual language understanding with regional knowledge. In *The Thirteenth International Conference on Learning Representations*.

Phillip Rust, Jonas Pfeiffer, Ivan Vulic, Sebastian Ruder, and Iryna Gurevych. 2021. 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, ACL/IJCNLP 2021*, pages 3118–3135. ACL.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 678–702. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual*

*Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4149–4158. Association for Computational Linguistics.

Alexey Tikhonov and Max Ryabinin. 2021. It's all in the heads: Using attention heads as a baseline for cross-lingual transfer in commonsense reasoning. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3534–3546. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 5998–6008.

Ada Wan. 2022. Fairness in representation for multilingual NLP: Insights from controlled experiments on conditional language modeling. In *International Conference on Learning Representations*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Trans. Assoc. Comput. Linguistics*, 10:291–306.

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

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3687–3692, Hong Kong, China. Association for Computational Linguistics.

Shiyue Zhang, Vishrav Chaudhary, Naman Goyal, James Cross, Guillaume Wenzek, Mohit Bansal, and Francisco Guzmán. 2022. How robust is neural machine translation to language imbalance in multilingual tokenizer training? In *Proceedings of the 15th biennial conference of the Association for Machine Translation in the Americas (Volume 1: Research Track), AMTA 2022, Orlando, USA, September 12-16, 2022*, pages 97–116. Association for Machine Translation in the Americas.

Wenxuan Zhang, Mahani Aljunied, Chang Gao, Yew Ken Chia, and Lidong Bing. 2023. M3exam: A multilingual, multimodal, multilevel benchmark for examining large language models. *Advances in Neural Information Processing Systems*, 36:5484–5505.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023a. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207, Toronto, Canada. Association for Computational Linguistics.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. 2023b. A formal perspective on byte-pair encoding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 598–614, Toronto, Canada. Association for Computational Linguistics.

## A Pseudocode

**Algorithm 1:** Algorithm for learning $\mathbf{m}$ using Classical BPE.

**Input:** Corpus $\mathcal{D}$; number of merges $K$
**Output:** Vocabulary $\mathcal{V}_K$; merge sequence $\mathbf{m}_K$

$\mathcal{V}_0 \leftarrow \mathcal{B}$
$\mathbf{m}_0 \leftarrow \langle\,\rangle$
**for** $k \leftarrow 1$ **to** $K$ **do**
  // Count all adjacent token pairs
  *Pairs* $\leftarrow \{\}$
  **foreach** *occurrence of consecutive tokens* $v\,v'$ *in* $\mathcal{D}$ *where* $v, v' \in \mathcal{V}_{k-1}$ **do**
    *Pairs*$[(v, v')] \leftarrow$ *Pairs*$[(v, v')] + 1$
  $(v^\star, v'^\star) \leftarrow \arg\max_{(v,v')}$ *Pairs*$[(v, v')]$
  $w^\star \leftarrow v^\star \circ v'^\star$
  // Update vocabulary and merge sequence
  $\mathcal{V}_k \leftarrow \mathcal{V}_{k-1} \cup \{w^\star\}$
  $\mathbf{m}_k \leftarrow \mathbf{m}_{k-1} + \langle (v^\star, v'^\star) \rangle$
  // Replace all occurrences in corpus
  **foreach** *occurrence of* $v^\star\,v'^\star$ *in* $\mathcal{D}$ **do**
    Replace $v^\star\,v'^\star$ with $w^\star$
**return** $\mathcal{V}_K$, $\mathbf{m}_K$

**Algorithm 2:** Algorithm for learning $\mathbf{m}$ using Parity-aware Byte Pair Encoding with separate training and development sets.

**Input:** $\{\mathcal{D}_\ell\}_{\ell\in\mathcal{L}}$ (multilingual training corpus); $\{\mathcal{D}_\ell^{\text{dev}}\}_{\ell\in\mathcal{L}}$ (multilingual development corpus); $K$ (number of merges)
**Output:** $\mathcal{V}_K$ (vocabulary); $\mathbf{m}_K$ (merge list)

$\mathcal{V}_0 \leftarrow \mathcal{B}$; $\mathbf{m}_0 \leftarrow \langle\,\rangle$
**for** $k \leftarrow 1$ **to** $K$ **do**
  // Calculate compression rate for each language
  **foreach** *language* $\ell \in \mathcal{L}$ **do**
    $\mathrm{CR}(\mathcal{D}_\ell^{\text{dev}}, \tau_{\mathbf{m}_{<k}}) \leftarrow \frac{\sum_{\mathbf{b}\in\mathcal{D}_\ell^{\text{dev}}} |\mathbf{b}|_u}{\sum_{\mathbf{b}\in\mathcal{D}_\ell^{\text{dev}}} |\tau_{\mathbf{m}_{<k}}(\mathbf{b})|}$
  $\ell^\star \leftarrow \arg\min_{\ell\in\mathcal{L}}\ \mathrm{CR}(\mathcal{D}_\ell^{\text{dev}}, \tau_{\mathbf{m}_{<k}})$
  // Consider token pairs only in $\mathcal{D}_{\ell^\star}$
  *Pairs* $\leftarrow \{\}$
  **foreach** *occurrence of consecutive tokens* $v\,v'$ *in* $\mathcal{D}_{\ell^\star}$ *where* $v, v' \in \mathcal{V}_{k-1}$ **do**
    *Pairs*$[(v, v')] \leftarrow$ *Pairs*$[(v, v')] + 1$
  $(v^\star, v'^\star) \leftarrow \arg\max_{(v,v')}$ *Pairs*$[(v, v')]$
  $w^\star \leftarrow v^\star \circ v'^\star$
  // Update vocabulary and merge list
  $\mathcal{V}_k \leftarrow \mathcal{V}_{k-1} \cup \{w^\star\}$
  $\mathbf{m}_k \leftarrow \mathbf{m}_{<k} + \langle (v^\star, v'^\star) \rangle$
  // Apply merge across all languages
  **foreach** *language* $\ell \in \mathcal{L}$ **do**
    **foreach** *occurrence of* $v^\star\,v'^\star$ *in* $\mathcal{D}_\ell$ *and* $\mathcal{D}_\ell^{\text{dev}}$ **do**
      Replace $v^\star\,v'^\star$ with $w^\star$
**return** $\mathcal{V}_K$, $\mathbf{m}_K$

## B Intrinsic Tokenizer Evaluation Metrics

We provide detailed descriptions of the intrinsic tokenizer metrics used in §5, grouped by the general tokenizer characteristic the metric aims to assess. Metric formulae are defined in terms of our definition of a tokenizer $T = (\mathcal{V}, \tau, \lrcorner)$ given in §2. For this tokenizer, we denote the empirical unigram frequency distribution of tokens $v \in \mathcal{V}$ as $X_T$, which is computed on our evaluation corpus.

### B.1 Vocabulary Usage

**Vocabulary Utilization and Type-Token Ratio.** Vocabulary utilization measures the proportion of a tokenizer's full vocabulary that is actively used when processing a given corpus. For tokenizer $T$ on corpus $\mathcal{D}$, we compute it as:

$$\text{VocabUtil}(T) = \frac{|\{v : v \in \tau(\mathbf{b}), \mathbf{b} \in \mathcal{D}\}|}{|\mathcal{V}|} \quad (7)$$

Here, the numerator counts the number of distinct tokens observed across the tokenization of all strings in the corpus. The type-token ratio quantifies lexical diversity by measuring the proportion of unique tokens (types) relative to the total number of tokens produced by a tokenizer:

$$\text{TTR}(T) = \frac{|\{v : v \in \tau(\mathbf{b}), \mathbf{b} \in \mathcal{D}\}|}{\sum_{\mathbf{b}\in\mathcal{D}} |\tau(\mathbf{b})|} \quad (8)$$

where $|\tau(\mathbf{b})|$ is the number of tokens produced by tokenizer $T$ for input $\mathbf{b}$. In words, the numerator counts distinct token types and the denominator counts total tokens across the corpus.

High vocabulary utilization and type-token ratio indicate efficient use of the learned vocabulary; low

values of these metrics for a particular language may suggest tokenizer bias, as only a small portion of the tokenizer's vocabulary is used/applicable for that language.

**Average Token Rank.** Average token rank (Limisiewicz et al., 2023) measures the typical position of tokens in a tokenized text within the frequency-ordered vocabulary. In more detail, we compute the rank of each token (denoted as $\text{rank}(v)$) in our unigram frequency distribution $X_T$; rank 1 corresponds to the most frequent token. We compute average token rank across tokens in the evaluation corpus as:

$$\text{AvgRank}(T) = \frac{\sum_{\mathbf{b}\in\mathcal{D}} \sum_{v\in\tau(\mathbf{b})} \text{rank}(v)}{\sum_{\mathbf{b}\in\mathcal{D}} |\tau(\mathbf{b})|} \quad (9)$$

This metric can be seen as another measure of the proportion of the vocabulary used by a tokenizer. Lower average ranks indicate that the tokenizer predominantly uses a small set of tokens, while higher averages suggest more diverse token usage, including rare vocabulary items. When computed per language (*i.e.*, when ranks are computed using the language's respective frequency distribution), systematic differences in average token rank across languages reveal vocabulary allocation bias.

### B.2 Information-theoretic Metrics

**Compression Rate.** We evaluate compression rate—as defined in eq. 1—across a parallel corpus. As discussed in §3.2, this enables us to use lines (documents) as our normalization unit. Recall that higher compression rates are generally desirable for computational efficiency in downstream tasks. In multilingual corpora, compression ratio disparities across languages indicate systematic tokenizer bias, where certain languages achieve better compression efficiency than others, potentially leading to unequal computational costs.

**Rényi Entropy.** We compute Rényi entropy of order $\alpha$ over the empirical unigram frequency distribution $X_T$ for a given tokenizer $T$ to capture different aspects of token distribution:

$$H_\alpha(X_T) = \frac{1}{1-\alpha} \log_2 \left( \sum_{v\in\mathcal{V}} p(v)^\alpha \right) \quad (10)$$

for $0\alpha < \infty$; at $\alpha = 1$, the definition of Shannon entropy is typically adopted. Rényi entropy provides a parametric family of measures that emphasize different aspects of the distribution: $H_1$ (Shannon entropy), $H_2$ (collision entropy), and $H_\infty$ (min-entropy). Rényi efficiency is Rényi entropy normalized by the size of the support, which is helpful for comparing tokenizers with different vocabulary sizes (Zouhar et al., 2023a). As all of our comparisons are between tokenizers of the same vocabulary size, we omit this normalization step and compare entropies directly.

### B.3 Morphological and Multilingual Fairness Metrics

**Fertility.** Fertility measures the average number of tokens produced per unit (word, character, or byte) by a tokenizer; the unit of interest for fertility is often the *word*, in which case, fertility quantifies how many tokens (on average) a word is broken up into. We use words as our normalization unit in our computations, as determined by the HuggingFace Whitespace Pretokenizer. We formally define tokenizer fertility for a given corpus $\mathcal{D}$ as:

$$\text{Fertility}(T) = \frac{\sum_{\mathbf{b}\in\mathcal{D}} |\tau(\mathbf{b})|}{\sum_{\mathbf{b}\in\mathcal{D}} |\mathbf{b}|_u} \quad (11)$$

This metric can give a sense for the computational efficiency imbued by a tokenizer, as well as for sequence length estimates for downstream modeling tasks.

**MorphScore.** MorphScore (Arnett et al., 2025) evaluates tokenizer quality through morpheme-level precision and recall, measuring how well tokenizers preserve morphological information during segmentation. We point the reader to the original work. Differences in cross-language MorphScore reveal how consistently a tokenizer's sub-token boundaries align with true morpheme boundaries. A higher score in one language than another indicates that the tokenizer preserves that language's morphological structure more faithfully. MorphScore provides a notion of both precision and recall (we point the reader to the original work for the exact description of the computation). Low precision indicates tokenizer oversegmentation; low recall is suggestive of *under* segmentation.

**Tokenizer Fairness Gini Coefficient.** We use an adaptation of the Gini coefficient—often used as a measure of economic inequality—to encapsulate tokenizer fairness across languages (Meister, 2025). Formally, let $c_1 \leq c_2 \leq \ldots \leq c_n$ be the "costs" under a given tokenizer $T$ for languages

$\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$. Here, we quantify cost as the average number of tokens it takes to encode the unit of interest (*e.g.*, a byte, word or line);[10] when using a parallel corpus, this can be cost per line (document), which controls for discrepancies between average character byte lengths across different scripts. The Gini coefficient for tokenizer $T$ is then:

$$\text{Gini}(T) = \frac{1}{n}\left(n + 1 - 2\frac{\sum_{i=1}^{n}(n+1-i)c_i}{\sum_{i=1}^{n} c_i}\right) \tag{12}$$

Values range from 0 (completely equal costs across languages) to 1 (maximum inequality). This metric condenses multilingual tokenizer fairness into a single number by measuring the degree of inequality in computational costs across languages; lower Gini coefficients indicate more equitable tokenizer compression across languages, while higher values suggest systematic bias toward certain languages.

# C Hyperparameter Selection

## C.1 Hybrid parity-aware BPE

For the hybrid parity-aware tokenizer, we set the number of parity-aware merges $K$ equal to the number of classical BPE merges $J$ (*i.e.*, $K = J$, with each set to either $64k$ or $128k$). This choice represents a midpoint between classical BPE ($J > 0, K = 0$) and fully parity-aware BPE ($J = 0, K > 0$). Alternative settings allow practitioners to trade off efficiency on high-resource languages (*e.g.*, English) against cross-lingual fairness. Because no single operating point is objectively optimal across use cases, we do not claim a universally optimal choice for this trade-off.

## C.2 Moving-Window Balancing

The moving-window balancing mechanism is designed to prevent degenerate cases in which a single language repeatedly consumes merge operations despite diminishing compression gains. The window size parameter ranges from 1 (enforcing uniform language selection across merges) to the number of languages $N$ (equivalent to no constraint). In a 30-language setting, this implies that even if compression improvements for a given language stagnate, it can consume at most approximately $1/15$ of merge operations. Increasing the window size causes the behavior to increasingly resemble the unbalanced setting.

[10] This is equivalent to fertility, or the inverse of the compression rate.

Because this mechanism primarily guards against pathological merge allocation rather than directly optimizing compression, we do not expect the window size to be a sensitive hyperparameter.

To validate this intuition, we conduct ablation experiments varying the window size in the moving-window balancing variant. Table 3 reports intrinsic tokenizer metrics for window sizes ranging from 50 to 200. Average compression rate, morphological plausibility, and other intrinsic metrics remain largely stable across window sizes. Empirically, the windowing mechanism prevents repeated selection of the same language caused by development–training set mismatch, without introducing sensitivity to the specific window size. These results suggest that the moving-window approach can be applied robustly without careful tuning.

## C.3 Parallel Development Set Size

We further study the effect of the size of the parallel development set used in cross-lingual compression rate comparisons for language selection (Eq. 7). Table 4 reports results for development set sizes of 100, 300, and 1000 examples.

The results indicate that relatively small development sets suffice to capture most of the fairness improvements. A parallel development set of 100 examples achieves a 67% reduction in the Gini Inequality Coefficient relative to the classical baseline, compared to a 72% reduction when using the full 1000-example development set. Global compression rates remain essentially unchanged across settings, suggesting that incorporating a parallel development set is feasible without imposing substantial data or computational burdens.

# D Additional Results and Ablation Studies

In this section, we present the results of our ablation studies. Table 5 reports the intrinsic evaluation of tokenizers with a $128k$ vocabulary size on the (unbalanced) *60-lang* dataset. Table 6 shows the corresponding results for the (balanced) *30-lang* dataset, also with a $128k$ vocabulary size. Finally, Table 7 presents the intrinsic evaluation of tokenizers with a $256k$ vocabulary size on the (*unbalanced*) *30-lang* dataset. Together, these results demonstrate the effectiveness of Parity-aware BPE across different language settings, vocabulary sizes, and data distributions.

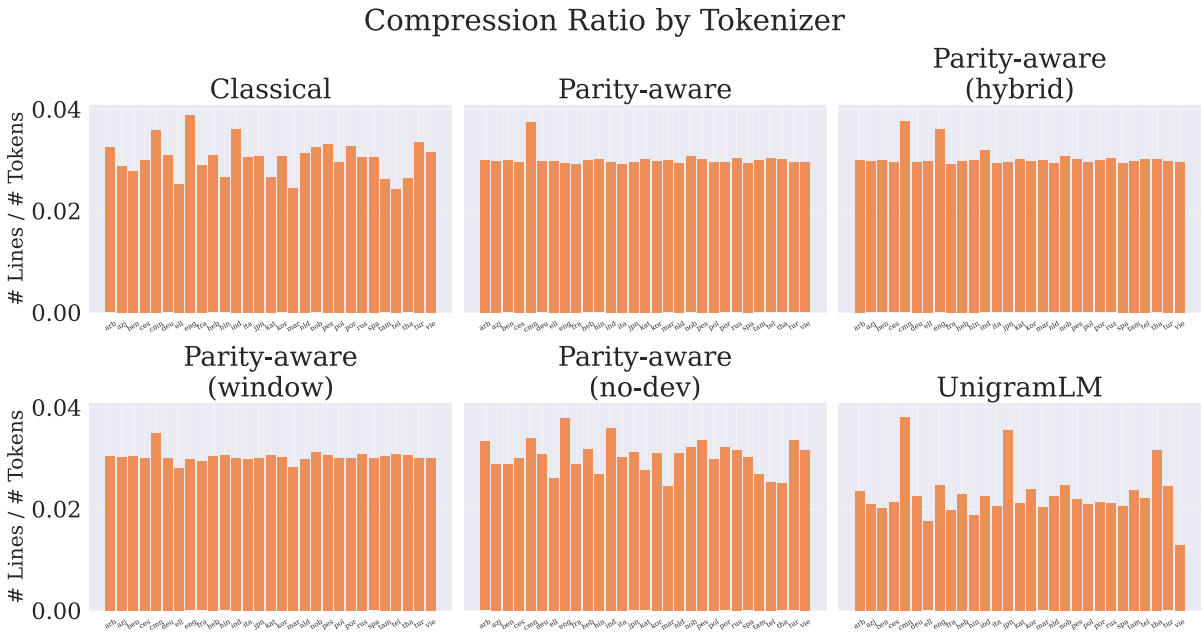


Figure 2: Compression rate of $128k$ tokenizers on the (*unbalanced*) *30-lang* per language.

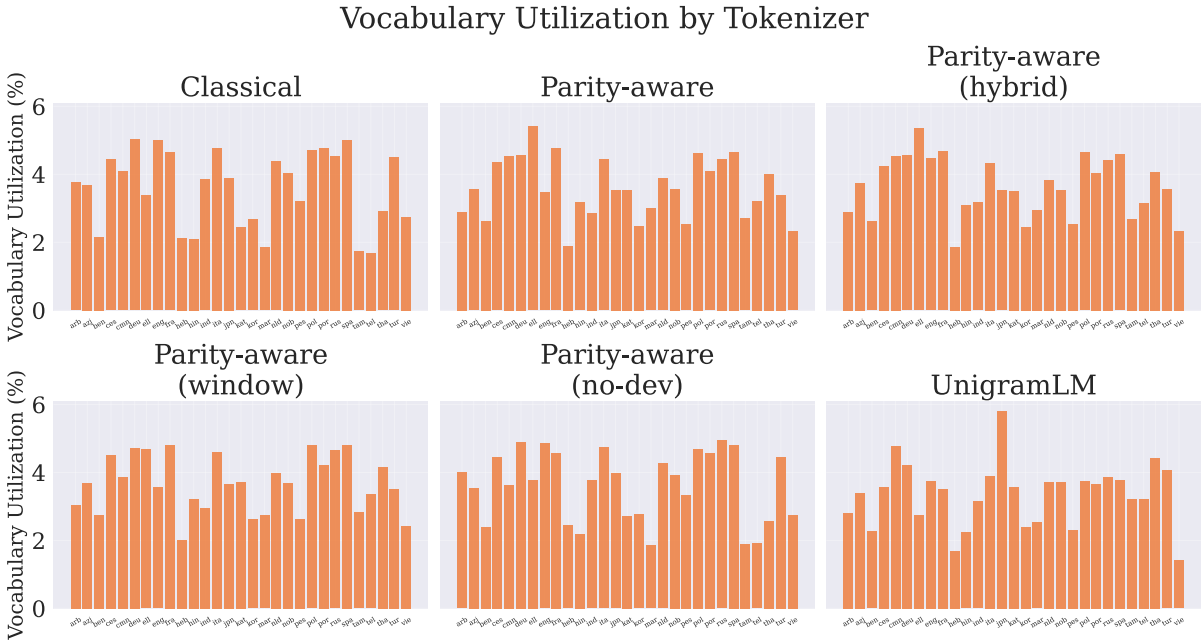


Figure 3: Vocabulary utilization of $128k$ tokenizers on the (*unbalanced*) *30-lang* per language.

| Tokenizer | Comp. Rate | TTR | Fertility | Rényi ($\alpha = 2.5$) | Gini | MorphScore |
|---|---|---|---|---|---|---|
| Classical | 0.0283 | 0.0706 | 4.508 | 7.56 | 0.061 | 0.412 |
| Parity-aware (hybrid) | 0.0284 | 0.0731 | 4.439 | 7.53 | 0.017 | 0.412 |
| Hybrid + window 50 | 0.0285 | 0.0751 | 4.444 | 7.52 | 0.020 | 0.412 |
| Hybrid + window 100 | 0.0286 | 0.0754 | 4.451 | 7.52 | 0.021 | 0.416 |
| Hybrid + window 150 | 0.0286 | 0.0755 | 4.453 | 7.52 | 0.022 | 0.418 |
| Hybrid + window 200 | 0.0286 | 0.0754 | 4.451 | 7.52 | 0.021 | 0.416 |

Table 3: Intrinsic tokenizer evaluation metrics for ablations over the moving-window size. Results indicate minimal sensitivity to the window size parameter.

| Tokenizer | Comp. Rate | TTR | Fertility | Rényi ($\alpha = 2.5$) | Gini | MorphScore |
|---|---|---|---|---|---|---|
| Classical | 0.0283 | 0.0706 | 4.508 | 7.56 | 0.061 | 0.412 |
| Hybrid (dev=100) | 0.0283 | 0.0724 | 4.400 | 7.54 | 0.020 | 0.411 |
| Hybrid (dev=300) | 0.0283 | 0.0727 | 4.426 | 7.54 | 0.018 | 0.412 |
| Hybrid (dev=1000) | 0.0284 | 0.0731 | 4.439 | 7.53 | 0.017 | 0.412 |

Table 4: Intrinsic tokenizer evaluation metrics for ablations over the size of the parallel development set. Smaller development sets retain most fairness gains.

**Training Data Distribution.** To assess the sensitivity of the Parity-aware algorithm to training data distribution, we also analyze results for $128k$ tokenizers trained on the *balanced* version of the dataset. The results in Table 6 indicate that parity-aware BPE tokenizers perform similarly to Classical BPE across most metrics. However, in terms of fertility, Classical BPE outperforms the parity-aware variants. This suggests that parity-aware tokenizers are particularly beneficial in unbalanced settings, where low-resource languages are more disadvantaged, whereas in balanced scenarios their advantage diminishes. We also interestingly see in Fig. 4—again for the (*balanced*) *30-lang* setting—that parity-aware tokenizers yield the largest absolute increases in vocabulary utilization for high-resource languages. Low- and medium-resource languages also improve, though to a smaller extent. One logical conclusion from this result is that the effect of parity-aware tokenizer is better described as balancing utilization across languages rather than directly compensating for data scarcity.

**Script Analysis.** Fig. 5 illustrates vocabulary utilization across languages for $128k$ tokenizers on the (unbalanced) *30-lang* dataset. For Latin, Arabic, Hebrew, and Cyrillic scripts, the Parity-aware BPE (no-dev) variant outperforms all other parity-aware BPE versions, with Classical BPE ranking second. In contrast, for the CJK scripts, Classical BPE leads, while all parity-aware BPE variants perform similarly and closely follow. For the remaining scripts, the base Parity-aware BPE or the window-balanced variant significantly outperform other tokenizers. These findings suggest that different scripts benefit differently from parity-aware BPE approaches.

**Language Model Perplexities.** We report language model perplexities on the FineWeb2 validation set in Fig. 6. Results are shown per language. We see a noticeably larger cross-lingual spread in perplexity for language models trained using the Classical BPE tokenizer than for those trained using Parity-aware variants. The Parity-aware tokenizers seem to eliminate the long tail present under Classical BPE while maintaining comparable mean perplexity across languages. Note that we normalize by number of bytes in the text rather than by number of tokens to account for differences in tokenization lengths.

## E Balancing the tokenizer training set

Prior work has addressed representational imbalance in subword tokenization primarily through corpus-level reweighting or balancing strategies, such as equalizing the number of documents or bytes per language prior to training a BPE or unigram tokenizer. While such approaches ensure that low-resource languages contribute to the learned vocabulary, they optimize the tokenizer with respect to an artificial distribution that differs from the natural data distribution encountered during downstream training and inference. This mismatch can result in inefficient allocation of vocabulary capacity, over-representation of rare patterns, and increased tokenization length for high-frequency languages. Moreover, corpus balancing constitutes a coarse-grained intervention: the choice of balancing criterion substantially affects outcomes, and the

| Tokenizer | Type-Token Ratio | Fertility | Compression Rate | Rényi Entropy ($\alpha$=2.5) | Gini Coefficient | MorphScore Precision | MorphScore Recall |
|---|---|---|---|---|---|---|---|
| Classical | 0.0388 | 3.374 ± 0.027 | 0.0277 ± 0.0000 | 8.16 | 0.086 | 0.324 ± 0.031 | 0.407 ± 0.029 |
| Parity-aware | 0.0321 | 3.533 ± 0.029 | 0.0260 ± 0.0000 | **8.25** | **0.022** | 0.273 ± 0.030 | 0.379 ± 0.028 |
| Parity-aware (hybrid) | 0.0334 | 3.453 ± 0.028 | 0.0269 ± 0.0000 | 8.20 | 0.040 | 0.283 ± 0.030 | 0.379 ± 0.028 |
| Parity-aware (window) | 0.0392 | 3.438 ± 0.028 | 0.0270 ± 0.0000 | 8.17 | 0.030 | 0.305 ± 0.029 | 0.393 ± 0.028 |
| Parity-aware (window+hybrid) | **0.0409** | **3.362 ± 0.028** | **0.0278 ± 0.0000** | 8.12 | 0.044 | 0.317 ± 0.029 | 0.400 ± 0.028 |
| Parity-aware (no-dev) | 0.0401 | 3.412 ± 0.027 | 0.0276 ± 0.0000 | 8.16 | 0.080 | **0.334 ± 0.031** | **0.418 ± 0.029** |

Table 5: Intrinsic evaluation of $128k$ tokenizers on the (*unbalanced*) ***60-lang*** dataset. Values are global statistics, except for MorphScore, which is macro-averaged across available languages.

| Tokenizer | Type-Token Ratio | Fertility | Compression Rate | Rényi Entropy ($\alpha$=2.5) | Gini Coefficient | MorphScore Precision | MorphScore Recall |
|---|---|---|---|---|---|---|---|
| Classical | 0.0780 | **4.175 ± 0.049** | 0.0307 ± 0.0001 | 8.09 | 0.050 | 0.409 ± 0.048 | 0.454 ± 0.046 |
| Parity-aware | 0.0765 | 4.207 ± 0.049 | 0.0300 ± 0.0001 | **8.12** | **0.011** | 0.405 ± 0.052 | 0.455 ± 0.049 |
| Parity-aware (hybrid) | 0.0767 | 4.192 ± 0.049 | 0.0303 ± 0.0001 | 8.11 | 0.016 | 0.404 ± 0.050 | 0.448 ± 0.048 |
| Parity-aware (window) | 0.0787 | 4.222 ± 0.050 | 0.0302 ± 0.0001 | 8.11 | 0.013 | 0.407 ± 0.050 | 0.455 ± 0.047 |
| Parity-aware (window+hybrid) | 0.0794 | 4.177 ± 0.049 | 0.0305 ± 0.0001 | 8.09 | 0.020 | 0.413 ± 0.049 | 0.457 ± 0.047 |
| Parity-aware (no-dev) | **0.0802** | 4.234 ± 0.050 | 0.0306 ± 0.0001 | 8.09 | 0.047 | **0.418 ± 0.048** | **0.463 ± 0.046** |
| Parity-aware (hybrid+no-dev) | 0.0800 | 4.231 ± 0.050 | **0.0307 ± 0.0001** | 8.08 | 0.048 | 0.415 ± 0.048 | 0.458 ± 0.046 |

Table 6: Intrinsic evaluation of $128k$ tokenizers on the (*balanced*) ***30-lang*** dataset. Values are global statistics, except for MorphScore, which is macro-averaged across available languages.

| Tokenizer | Type-Token Ratio | Fertility | Compression Rate | Rényi Entropy ($\alpha$=2.5) | Gini Coefficient | MorphScore Precision | MorphScore Recall |
|---|---|---|---|---|---|---|---|
| Classical | 0.1239 | 3.767 ± 0.044 | 0.0340 ± 0.0001 | 7.85 | 0.052 | 0.515 ± 0.053 | 0.545 ± 0.051 |
| Parity-aware | 0.1205 | 3.809 ± 0.045 | 0.0334 ± 0.0001 | **7.87** | **0.010** | 0.498 ± 0.053 | 0.533 ± 0.051 |
| Parity-aware (hybrid) | 0.1212 | 3.803 ± 0.045 | 0.0336 ± 0.0001 | 7.86 | 0.012 | 0.506 ± 0.053 | 0.538 ± 0.051 |
| Parity-aware (window) | 0.1268 | 3.781 ± 0.045 | 0.0338 ± 0.0001 | 7.84 | 0.013 | 0.510 ± 0.052 | 0.543 ± 0.050 |
| Parity-aware (window+hybrid) | 0.1275 | 3.772 ± 0.045 | 0.0340 ± 0.0001 | 7.83 | 0.017 | 0.518 ± 0.052 | 0.548 ± 0.051 |
| Parity-aware (no-dev) | 0.1272 | 3.799 ± 0.044 | 0.0341 ± 0.0001 | 7.84 | 0.050 | 0.531 ± 0.052 | **0.559 ± 0.051** |
| Parity-aware (hybrid+no-dev) | 0.1271 | 3.797 ± 0.044 | 0.0341 ± 0.0001 | 7.84 | 0.050 | **0.531 ± 0.052** | 0.559 ± 0.051 |

Table 7: Intrinsic evaluation of $256k$ tokenizers on the (*unbalanced*) ***30-lang*** dataset. Values are global statistics, except for MorphScore, which is macro-averaged across available languages.

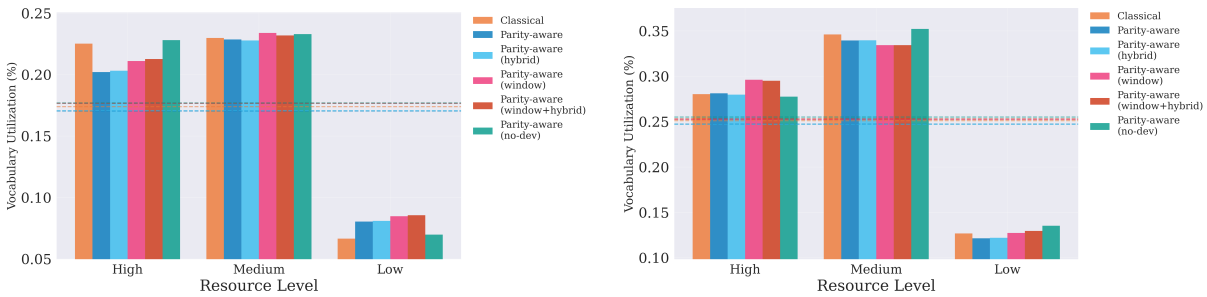


Figure 4: Vocabulary utilization grouped by language resource levels for the $256k$ tokenizer trained on the (*unbalanced*) *30-lang* dataset (left) and $128k$ tokenizer trained on the (*balanced*) *30-lang* dataset (right).

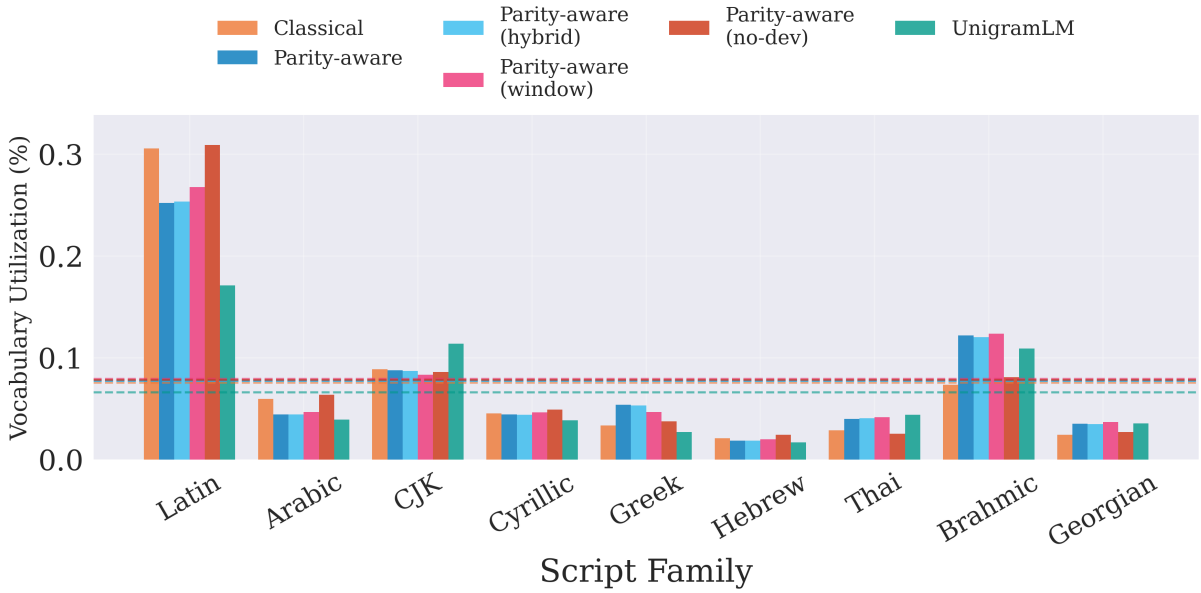


Figure 5: Vocabulary utilization across language scripts for $128k$ tokenizers on the (*unbalanced*) *30-lang* dataset.

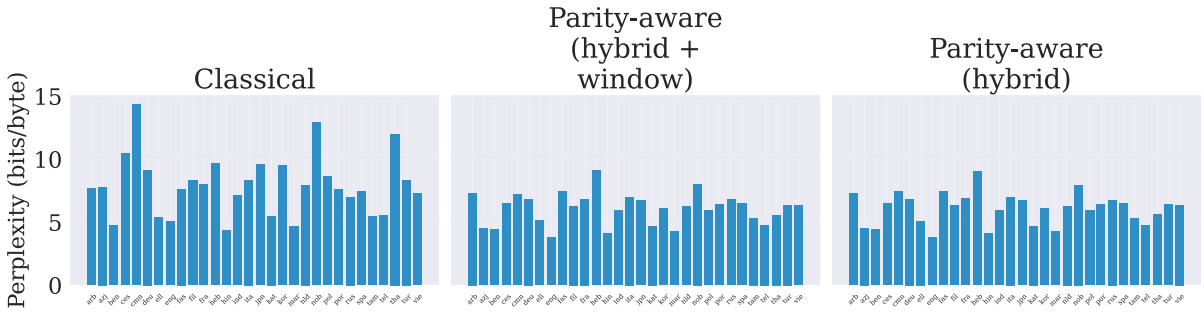


Figure 6: Per-language perplexities normalized by byte of language models trained using the specified tokenizer trained on the (*unbalanced*) $128k$ *30-lang* dataset. Results are computed on the language model validation set at the final checkpoint (see §F for language model details).

approach does not account for heterogeneity within groups or for the local utility of individual merge operations.

Parity-aware BPE instead operates on the unaltered corpus distribution and introduces language-level constraints directly into the merge selection process. By augmenting merge scores with parity-aware penalties or budgets, this approach enables fine-grained control over how subword units are allocated across languages, while preserving the efficiency benefits of frequency-driven merging. In contrast to the balancing approach, parity-aware merges maintain alignment with the downstream data distribution and typically yield lower average tokenization costs at inference, while mitigating systematic over-fragmentation of underrepresented languages. Overall, parity-aware BPE offers a principled mechanism for addressing tokenizer bias at the level of individual merge decisions, complementing existing corpus-level mitigation strategies.

# F Language Model Training

Here we provide details about the language models used for evaluating extrinsic tokenizer metrics.

## F.1 Model Architecture

Our experiments focus on models with 3 billion parameters (3B), following the LLaMA architecture (Touvron et al., 2023). Model size is determined by adjusting the number of layers, hidden dimensions, and attention heads.

## F.2 Training Hyperparameters

We train our models using SwissAI's fork of Hugging Face's Nanotron repository.[11] The key training hyperparameters are as follows:

- **Learning Rate.** We use a learning rate of $8 \times 10^{-4}$ with linear warmup over the first 4% of training. A "1-sqrt" decay schedule (Hägele et al., 2024) is applied during the final 20%.
- **Optimizer.** All experiments use AdamW with $\beta = (0.9, 0.95)$ (Loshchilov and Hutter, 2019).
- **Weight Decay.** We set the weight decay parameter to $\lambda = 0.1$ for regularization.
- **Batch Size.** The micro-batch size is fixed at 5 across all runs.

[11]The codebase: https://github.com/swiss-ai/nanotron-multilingual

## F.3 Hardware Setup

Training is performed on a large-scale cluster. Each node is equipped with 4 NVIDIA Grace-Hopper H100 GPUs (96 GB memory each). The 3B models are trained on 64 nodes (256 GPUs in total), requiring approximately 18 hours per 100B tokens. This corresponds to a global batch size of 640 examples.

## F.4 Sampling Methods

Let $\mathcal{L}$ be the set of languages in the dataset, and let $\pi^{\text{natural}} \in \Delta_{|\mathcal{L}|}$ represent the natural distribution of these languages, defined as:

$$\pi_l^{\text{natural}} = \frac{\omega_l}{\sum_{l' \in \mathcal{L}} \omega_{l'}}$$

where $\omega_l$ denotes the number of words (or tokens) for language $l$ in the dataset. In this work, we use the number of words as a proxy for language frequency, a common practice when presenting statistics for highly multilingual datasets (Penedo et al., 2025). We use Temperature Sampling which is defined as:
This method adjusts the natural distribution using a temperature parameter $\tau$ to create a less skewed distribution:

$$\pi_l^{\text{temp},\tau} = \frac{\omega_l^{1/\tau}}{\sum_{l' \in \mathcal{L}} \omega_{l'}^{1/\tau}}$$

By tuning $\tau$, the distribution can be shifted towards uniformity, thereby reducing imbalance among languages.

# G Downstream Benchmark Evaluation

We evaluate our models using HuggingFace's Lighteval codebase (Habib et al., 2023).

## G.1 Benchmarks

To assess multilingual performance, we select twelve widely used benchmarks that span diverse downstream tasks, including reading comprehension, commonsense reasoning, semantic similarity, and knowledge-based evaluation:

- **Belebele**: A multilingual reading comprehension dataset containing passages and corresponding questions in many languages. It evaluates models' ability to understand text and answer related questions (Bandarkar et al., 2024).

- **mTruthfulQA**: A multilingual extension of TruthfulQA, designed to measure whether models generate accurate and non-misleading answers across a broad range of questions (Lin et al., 2022a; Lai et al., 2023).

- **PAWS-X**: A multilingual paraphrase identification benchmark that extends the original PAWS dataset, providing sentence pairs annotated for semantic equivalence (Yang et al., 2019).

- **XCodah**: A multilingual adaptation of CODAH for adversarially-authored commonsense reasoning tasks, testing robustness in natural language understanding (Lin et al., 2021; Chen et al., 2019).

- **XCSQA**: A multilingual version of CommonsenseQA, consisting of multiple-choice questions that require reasoning about everyday concepts and their relations (Lin et al., 2021; Talmor et al., 2019).

- **XNLI**: A cross-lingual natural language inference benchmark, evaluating whether models can perform entailment, contradiction, and neutrality classification across multiple languages (Conneau et al., 2018).

- **XStoryCloze**: A multilingual extension of the StoryCloze Test, where models must choose the most coherent ending to short narratives, testing story comprehension and commonsense reasoning (Mostafazadeh et al., 2017; Lin et al., 2022b).

- **XWinogrande**: A multilingual version of WinoGrande, containing sentences with ambiguous pronouns. It measures models' ability to resolve coreference using contextual and commonsense cues (Sakaguchi et al., 2021; Muennighoff et al., 2023; Tikhonov and Ryabinin, 2021).

- **MMMLU**: A multilingual adaptation of MMLU, evaluating model performance across a wide spectrum of tasks and domains (Hendrycks et al., 2021; Lai et al., 2023).

- **INCLUDE**: A large-scale benchmark covering 44 languages, designed to evaluate multilingual LLMs in realistic language environments with a focus on knowledge and reasoning (Romanou et al., 2025).

- **Exams**: A benchmark of standardized test questions across subjects and educational levels, used to assess reasoning and problem-solving abilities in exam-like conditions (Hardalov et al., 2020).

- **M3Exams**: A multilingual exam-style benchmark that extends Exams across different languages, subjects, and difficulty levels (Zhang et al., 2023).

### G.2 Score Aggregations

We aggregate benchmark results to compute a language-specific score for each model. Let $\mathcal{T}_l$ be the set of benchmarks (or tasks) containing a split for language $l$. The aggregated score for a model $m$ per language $l$ is defined as:

$$s_l^m = \frac{1}{|\mathcal{T}_l|} \sum_{t \in \mathcal{T}_l} s_{t,l}^m$$

where $s_l^m$ is the score of a model $m$ on the split $l$ of a task $t$ To mitigate biases arising from varying numbers of benchmarks per language, we compute a language-specific random baseline $\zeta_l$. This baseline helps assess whether a given aggregated score significantly outperforms random predictions. Specifically, we calculate the random baseline for each language as the average of the individual random baselines across all tasks that include language $l$:

$$\zeta_l = \frac{1}{|\mathcal{T}_l|} \sum_{t \in \mathcal{T}_l} \zeta_t$$

| Language | Language Family | Script | Resource Level | 30-lang | 60-lang |
|---|---|---|---|---|---|
| English | Indo-European (Germanic) | Latin | High | ✓ | ✓ |
| German | Indo-European (Germanic) | Latin | High | ✓ | ✓ |
| French | Indo-European (Romance) | Latin | High | ✓ | ✓ |
| Italian | Indo-European (Romance) | Latin | High | ✓ | ✓ |
| Russian | Indo-European (Slavic) | Cyrillic | High | ✓ | ✓ |
| Spanish | Indo-European (Romance) | Latin | High | ✓ | ✓ |
| Japanese | Japonic | Kanji & Kana (CJK) | Medium | ✓ | ✓ |
| Polish | Indo-European (Slavic) | Latin | Medium | ✓ | ✓ |
| Portuguese | Indo-European (Romance) | Latin | Medium | ✓ | ✓ |
| Vietnamese | Austroasiatic | Latin | Medium | ✓ | ✓ |
| Turkish | Turkic | Latin | Medium | ✓ | ✓ |
| Dutch | Indo-European (Germanic) | Latin | High | ✓ | ✓ |
| Indonesian | Austronesian | Latin | Medium | ✓ | ✓ |
| Arabic | Afro-Asiatic (Semitic) | Perso-Arabic | Medium | ✓ | ✓ |
| Czech | Indo-European (Slavic) | Latin | Medium | ✓ | ✓ |
| Persian (Farsi) | Indo-European (Iranian) | Perso-Arabic | Medium | ✓ | ✓ |
| Greek | Indo-European (Hellenic) | Greek | Medium | ✓ | ✓ |
| Chinese (Mandarin) | Sino-Tibetan | Hanzi (CJK) | Medium | ✓ | ✓ |
| Hindi | Indo-European (Indo-Aryan) | Devanagari (Brahmic) | Medium | ✓ | ✓ |
| Korean | Koreanic | Hangugeo (CJK) | Medium | ✓ | ✓ |
| Thai | Kra–Dai (Tai) | Thai | Medium | ✓ | ✓ |
| Hebrew | Afro-Asiatic (Semitic) | Hebrew | Medium | ✓ | ✓ |
| Bengali | Indo-European (Indo-Aryan) | Bengali (Brahmic) | Medium | ✓ | ✓ |
| Tamil | Dravidian (Brahmic) | Tamil | Low | ✓ | ✓ |
| Georgian | Kartvelian | Georgian | Low | ✓ | ✓ |
| Marathi | Indo-European (Indo-Aryan) | Devanagari (Brahmic) | Medium | ✓ | ✓ |
| Filipino | Austronesian | Latin | Low | ✓ | ✓ |
| Telugu | Dravidian | Telugu (Brahmic) | Low | ✓ | ✓ |
| Norwegian | Indo-European (Germanic) | Latin | Medium | ✓ | ✓ |
| North Azerbaijani | Turkic | Latin | Low | ✓ | ✓ |
| Swedish | Indo-European (Germanic) | Latin | Medium | - | ✓ |
| Romanian | Indo-European (Romance) | Latin | Medium | - | ✓ |
| Ukrainian | Indo-European (Slavic) | Cyrillic | Medium | - | ✓ |
| Hungarian | Uralic (Ugric) | Latin | Medium | - | ✓ |
| Danish | Indo-European (Germanic) | Latin | Medium | - | ✓ |
| Finnish | Uralic (Finnic) | Latin | Medium | - | ✓ |
| Bulgarian | Indo-European (Slavic) | Cyrillic | Low | - | ✓ |
| Slovak | Indo-European (Slavic) | Latin | Low | - | ✓ |
| Catalan | Indo-European (Romance) | Latin | Low | - | ✓ |
| Malay | Austronesian | Latin | Low | - | ✓ |
| Urdu | Indo-European (Indo-Aryan) | Perso-Arabic | Low | - | ✓ |
| Belarusian | Indo-European (Slavic) | Cyrillic | Medium | - | ✓ |
| Basque | Language Isolate | Latin | Low | - | ✓ |
| Tajik | Indo-European (Iranian) | Cyrillic | Medium | - | ✓ |
| Sotho (Sesotho) | Niger–Congo (Bantu) | Latin | Low | - | ✓ |
| Yoruba | Niger–Congo | Latin | Low | - | ✓ |
| Swahili | Niger-Congo (Bantu) | Latin | Low | - | ✓ |
| Estonian | Uralic (Finnic) | Latin | Low | - | ✓ |
| Latvian | Indo-European (Slavic) | Latin | Low | - | ✓ |
| Galician | Indo-European (Romance) | Latin | Low | - | ✓ |
| Welsh | Indo-European (Celtic) | Latin | Low | - | ✓ |
| Albanian | Indo-European | Latin | Low | - | ✓ |
| Macedonian | Indo-European (Slavic) | Cyrillic | Low | - | ✓ |
| Malayalam | Dravidian | Malayalam (Brahmic) | Low | - | ✓ |
| Burmese | Sino-Tibetan | Mon–Burmese | Low | - | ✓ |
| Gujarati | Indo-European (Indo-Aryan) | Gujarati (Brahmic) | Low | - | ✓ |
| Afrikaans | Indo-European (Germanic) | Latin | Low | - | ✓ |
| Hawaiian | Austronesian | Latin | Low | - | ✓ |
| Northern Uzbek | Turkic | Latin | Low | - | ✓ |

Table 8: Details on the languages used to train and evaluate tokenizers.

| **Language** | INCLUDE | Belebele | Exams | M3Exam | MMMLU | mTruthfulQA | PAWS-X | XCodah | XCOPA | XCSQA | XNLI | XStoryCloze | XWinoGrande |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| English | - | ✓ | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Chinese | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Vietnamese | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ | - | - |
| Arabic | ✓ | ✓ | ✓ | - | ✓ | ✓ | - | ✓ | - | ✓ | ✓ | ✓ | - |
| German | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ | - | ✓ | - | - | - |
| Spanish | ✓ | ✓ | ✓ | - | ✓ | ✓ | - | ✓ | - | ✓ | ✓ | ✓ | - |
| French | ✓ | ✓ | - | - | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | - | ✓ |
| Portuguese | ✓ | ✓ | ✓ | - | ✓ | ✓ | - | ✓ | - | ✓ | - | - | ✓ |
| Hindi | ✓ | ✓ | - | - | ✓ | ✓ | - | ✓ | - | ✓ | ✓ | ✓ | - |
| Russian | ✓ | ✓ | - | - | ✓ | ✓ | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Indonesian | ✓ | - | - | - | ✓ | ✓ | - | - | ✓ | - | - | ✓ | - |
| Italian | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | ✓ | ✓ | ✓ | - | - | - |
| Japanese | ✓ | - | - | - | - | - | ✓ | ✓ | - | ✓ | - | - | ✓ |
| Swahili | - | ✓ | - | ✓ | - | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | - |
| Tamil | ✓ | - | - | - | - | - | - | - | ✓ | - | - | - | - |
| Telugu | ✓ | ✓ | - | - | ✓ | ✓ | - | - | - | - | - | ✓ | - |
| Thai | - | ✓ | - | ✓ | - | - | - | - | ✓ | - | ✓ | - | - |
| Basque | ✓ | - | - | - | ✓ | ✓ | - | - | - | - | - | ✓ | - |
| Turkish | ✓ | ✓ | ✓ | - | - | - | - | - | ✓ | - | ✓ | - | - |
| Bulgarian | ✓ | ✓ | ✓ | - | - | - | - | - | - | - | ✓ | - | - |
| Albanian | ✓ | ✓ | ✓ | - | - | - | - | - | - | - | - | - | - |
| Polish | ✓ | ✓ | - | - | - | - | - | ✓ | - | - | - | - | - |
| Bengali | ✓ | - | - | - | ✓ | ✓ | - | - | - | - | - | - | - |
| Serbian | ✓ | - | ✓ | - | - | ✓ | - | - | - | - | - | - | - |
| Estonian | ✓ | - | - | - | - | - | - | - | ✓ | - | - | - | - |
| Macedonian | ✓ | ✓ | - | - | - | - | - | - | - | - | - | - | - |
| Lithuanian | ✓ | - | ✓ | - | - | - | - | - | - | - | - | - | - |
| Greek | ✓ | - | - | - | - | - | - | - | - | - | ✓ | - | - |
| Urdu | ✓ | - | - | - | - | - | - | - | - | - | ✓ | - | - |
| Catalan | - | - | - | - | ✓ | ✓ | - | - | - | - | - | - | - |
| Persian | ✓ | - | - | - | - | - | - | - | - | - | - | - | - |
| Finnish | ✓ | - | - | - | - | - | - | - | - | - | - | - | - |
| Korean | ✓ | - | - | - | - | - | - | - | - | - | - | - | - |
| Quechua | - | - | - | - | - | - | - | - | ✓ | - | - | - | - |
| Haitian Creole | - | - | - | - | - | - | - | - | ✓ | - | - | - | - |
| Malay | - | - | - | - | - | - | - | - | - | - | - | ✓ | - |

Table 9: Coverage of downstream benchmarks across languages.