Phonemes to the Rescue: Multilingual Tokenization Based on International Phonetic Alphabet

Milan Miletić
University of Amsterdam
m.miletic@uva.nl

Julie Kallini
Stanford University
kallini@stanford.edu

Ekaterina Shutova
University of Amsterdam
e.shutova@uva.nl

Abstract

Multilingual language models often exhibit performance disparities across languages that can arise as early as the tokenization stage. Widely-used subword tokenization approaches favor high-resource languages, and tokenizer-free methods still yield longer sequences for scripts with a higher bytes-per-character ratio. To address these shortcomings, we propose to use the International Phonetic Alphabet (IPA) as a language-agnostic input representation for multilingual tokenizers. IPA provides a compact symbol inventory, greater cross-lingual character overlap, and a more balanced byte-per-character distribution across languages. We train matched pairs of text vs. IPA subword tokenizers across 24 languages and 14 scripts and demonstrate that IPA tokenizers consistently improve tokenization quality, especially for non-Latin scripts, and generalize more effectively to unseen languages and scripts.

github.com/Mikki99/ipa-tokenization

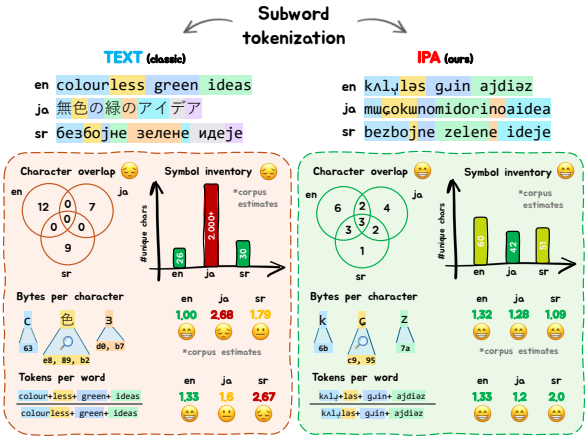


Figure 1: Benefits of IPA for multilingual tokenization. We show how our trained Text and IPA tokenizers segment the same input sequence in English (en), Japanese (ja), and Serbian (sr). Switching to IPA maps all languages into a shared phonetic alphabet, which (i) increases cross-lingual character overlap, (ii) reduces the symbol inventory needed to represent the corpus, and (iii) mitigates byte-per-character differences. Together, this results in better overall tokenization quality, as shown for instance by reduced tokens per word (what we later refer to as word fertility).

1 Introduction

Despite their growing global impact, multilingual language models (MLMs) continue to exhibit performance disparities across languages. Prior work shows that these differences originate as early as the tokenization stage, with direct consequences for downstream model performance (Petrov et al., 2023; Lotz et al., 2025; Arnett and Bergen, 2025; Rust et al., 2020). The main challenge of multilingual tokenization is how to represent a large number of diverse languages in a common fixed-size vocabulary. Currently predominant subword tokenization algorithms (BPE, Sennrich et al. 2016; UnigramLM, Kudo 2018) learn this vocabulary by optimizing objectives defined over corpus statistics. By design, this process favors languages that are more prevalent in the training set. As a result, underrepresented languages, especially those with rich morphologies or those using non-Latin scripts, often require more tokens for encoding the same semantic content (Ahia et al., 2023). This leads to increased computational cost, slower inference times, and diminished fluency and output quality in those languages (Qin et al., 2025). A prominent alternative to subword tokenization in recent years is *tokenization-free* byte-level modeling, which represents each input byte as a token. While this approach avoids many of the issues associated with traditional tokenizers, the shift to byte tokens *alone* does not eliminate cross-lingual disparities: characters in some scripts require up to 4× as many bytes to encode than those in other scripts, leading to longer input sequences—and thus, higher computational costs—for those languages (Mielke et al., 2021; Limisiewicz et al., 2024). This indicates that tokenization quality across languages depends not

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 40323–40349
July 2-7, 2026

only on the tokenization method (or lack thereof), but also on the input encoding on which it operates.

Guided by this observation, in this work we consider an alternative input representation that is offered by the International Phonetic Alphabet (IPA, International Phonetic Association 1999) instead of using standard orthography. IPA maps languages into a shared (phonetic) alphabet, increasing character-level overlap across languages that otherwise use disjoint scripts. Because the symbol inventory is small ($\approx$ 200 characters), a fixed-size subword vocabulary can be used more efficiently, with less of its capacity wasted on script-specific variants of the same words or morphemes. Finally, IPA is encoded mostly with 1–2 byte UTF-8 characters, which mitigates script-driven differences in bytes-per-character and therefore reduces disparities in per-language encoding cost. An overview of IPA's benefits is illustrated in Fig. 1 and we provide empirical results to support these claims in Appendix I. These benefits inspired IPA's usage in NLP at various stages of the modeling pipeline, from pre-training tasks (Gale et al., 2023) to prompting (Nguyen et al., 2024), or even integrating it into multimodal pipelines (Matsuhira et al., 2023). However, to the best of our knowledge, we are the first to systematically evaluate the effects of IPA on multilingual tokenization quality across a diverse set of configurations and languages.

Concretely, we train subword tokenizers on multilingual data spanning 24 languages and 14 distinct scripts in two input formats—standard orthography and IPA—while varying the tokenization algorithm, vocabulary size, and data sampling strategy. We evaluate each tokenizer on a suite of intrinsic tokenization quality metrics measuring compression, token frequency distribution properties, vocabulary usage, and cross-lingual equity. We further train GPT-2 models on a set of selected tokenizers and evaluate them on two widely used multilingual downstream tasks: XNLI and PAWS-X. We find that IPA tokenizers consistently outperform standard text tokenizers on intrinsic metrics, under their respective best configurations, without sacrificing downstream task performance. Overall, our approach preserves MLM capabilities, while yielding more equitable treatment across languages.

2 Background and Related Work

Disparities in tokenization quality. Prior work has shown that standard multilingual subword tokenizers systematically treat languages inequitably (Petrov et al., 2023; Ahia et al., 2023; Ali et al., 2024, inter alia). Resulting frequency-based vocabularies allocate more capacity to high-resource languages, so that lower-resource languages are split into many more tokens per word. This is especially amplified in case of morphologically more complex languages, as well as those using non-Latin scripts. Petrov et al. (2023) quantify this disparity, showing a difference in the amount of tokens needed to encode the same semantic content of up to $15\times$ between languages. This over-segmentation has direct consequences, as the impacted languages will require much higher training and inference costs.[1]

These findings have inspired several recent works that aim to tackle these tokenization challenges. Petrov et al. (2023) suggest training monolingual tokenizers for each language and then merging them based on tokenization parity. Inheriting the idea of tokenization parity, Foroutan et al. (2025) introduce Parity-Aware BPE, optimizing the training objective in classic BPE such that merges are guided by per-language compression levels, instead of simple frequency. Feher et al. (2025) propose a dynamic tokenization approach, which merges frequently co-occurring subwords on the fly to reduce the token-count inflation. The main alternative to subword tokenization in recent years have been different flavors of byte-level approaches, which eliminate learned segmentation by treating each byte in the input as a separate token (Xue et al., 2022; Kallini et al., 2024; Yu et al., 2023, inter alia). However, cross-lingual disparities still remain even when working with byte tokens, as languages differ in the number of bytes-per-character, as well as the number of symbols needed to encode the same semantic content (Arnett et al., 2024). Recent work has sought to address this issue with more adaptive forms of byte-level tokenization. Ahia et al. (2024) introduce MAGNET, which integrates adaptive gradient-based tokenization into the model through script-specific predictors that place token boundaries so as to equalize segmentation granularity across scripts. Owodunni et al. (2025) propose FLEXITOKENS, introducing a more flexible training objective that allows the compression rate to adapt across languages and scripts, as well as across domains within the same language.

While our method could in principle be applied

[1] Lundin et al. (2025) refer to this effect as *token tax*, showing that doubling in tokens can increase training cost by a factor of four.

	Latn									Arab			Cyrl		Deva	Jpan	Hang	Hani	Thai	Laoo	Mymr	Grek	Hebr	Ethi
Stage	de	en	es	fi	fr	it	pl	sw	tr	ar	fa	ur	ru	sr	hi	ja	ko	zh	th	lo	my	el	he	am
Tokenizer training	✓	✓	-	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-
GPT-2 pre-training	✓	✓	-	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-
Intrinsic eval	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
XNLI	✓	✓	✓	-	✓	-	-	✓	✓	✓	-	✓	✓	-	✓	-	-	✓	✓	-	-	✓	-	-
PAWS-X	✓	✓	✓	-	✓	-	-	-	-	-	-	-	-	-	-	✓	✓	✓	-	-	-	-	-	-

Table 1: Language coverage at different stages of our experiments. Columns show languages (ISO 639-1 codes) grouped by script (ISO 15924 codes). We provide the language/script-to-ISO mapping in Appendix C.1. ✓ indicates that a language is included in the corresponding stage. For tokenizer and GPT-2 training we use a sample from the CulturaX dataset (Nguyen et al., 2023). For intrinsic evaluation we use WikiPron (Lee et al., 2020) and FLORES+ (NLLB Team et al., 2024). We perform downstream evaluation on XNLI (Conneau et al., 2018) and PAWS-X (Yang et al., 2019).

to byte-level approaches (see Appendix J), we focus on currently predominant subword tokenization in this work. We provide a summary of different tokenization methods in Appendix B.

IPA in Natural Language Processing. IPA has seen a growing range of applications in NLP in recent years, particularly in settings that require phonological awareness. Leong and Whitenack (2022) and Sohn and Mortensen (2025) show benefits of IPA representations for improved performance in named entity recognition (NER) tasks. Gale et al. (2023) introduce the BORT model, an extension of BART (Lewis et al., 2020) with added self-supervised pre-training pronunciation tasks by leveraging IPA representations. Nguyen et al. (2024) propose phonemic prompting as a way to enhance the multilingual capabilities of LLMs. Matsuhira et al. (2023) developed IPA-CLIP, by incorporating IPA representations into a multimodal setting, extending the CLIP model. Taken together, these studies highlight the versatility and effectiveness of IPA in enhancing cross-lingual generalization, representation learning, and multilingual performance. Most closely related to our work are Goriely et al. (2024), who also pre-train GPT-2 models on IPA input, testing both character-level and BPE tokenization. However, their study is limited to a monolingual setting (English) and does not analyze the impact of IPA on tokenization quality.

3 Data and Languages

3.1 Languages

Table 1 summarizes language coverage at each stage of our experiments. For tokenizer training (Section 4.3) and GPT-2 pre-training (Section 4.5) we consider 18 languages spanning 10 scripts. We evaluate the intrinsic tokenization quality (Section 4.4) on the same language set together with six additional languages (am, el, es, he, my, pl) that introduce four new scripts (Grek, Hebr, Ethi, Mymr) and allow us to test zero-shot generalization to languages/scripts unseen during tokenizer training. In total, this results in 24 languages and 14 scripts considered in this work.

Our language selection aimed to cover a diverse sample across orthographic, typological, and resource profiles. For instance, it includes high-resource Indo-European languages (e.g., en, de, fr, it, ru), as well as comparatively lower-resource languages (e.g., sw, lo, my, am), which are often underrepresented in multilingual pre-training pipelines. We also consider morphologically rich languages (e.g., fi, tr) which often exhibit worse tokenization quality (Raj et al., 2024) and those with more complex orthographies (e.g., zh, ja, ko, th, lo) which are typically more expensive to encode due to their Unicode ranges. We also include a language written in multiple scripts, ja, spanning Katakana, Hiragana, and Kanji (collectively labeled as Jpan).[2,3]

3.2 Datasets

CulturaX. For tokenizer training (Section 4.3) and GPT-2 pre-training (Section 4.5), we use CulturaX (Nguyen et al., 2023), a large-scale multilingual web corpus derived from filtered mC4 (Xue et al., 2021) and OSCAR (Suárez et al., 2019, 2020). We apply additional post-processing to remove residual non-linguistic artifacts and reduce cross-language contamination on our sample of languages (Appendix C.4). We sample a total of 1GB of

[2] Although, in our experiments we only consider Katakana and Hiragana, due to limitations in Epitran's Kanji support

[3] Serbian is also a is also a multi-script language (Latn and Cyrl), but our sample only contained Cyrillic text.

data distributed across our 18 languages using four strategies: (i) `byte-uniform`, allocating an equal number of bytes to each language; (ii) `semantic-uniform`, allocating equal semantic content by compensating for language-specific byte premiums (Arnett et al., 2024); (iii) `data-proportional`, sampling in proportion to each language's share in CulturaX; and (iv) `data-smoothed`, applying temperature sampling (α=0.3) to upweight lower-resource languages (as done in mT5, Xue et al. (2021); mBERT, Devlin (2018); XLM-R, (Conneau et al., 2020)). The first two strategies provide a *balanced* sample of languages (under different notions of balance), while the latter two result in *unbalanced*, but arguably more realistic distributions in practice. More details and distribution visualizations are in Appendix C.2.

WikiPron and FLORES+. For evaluating intrinsic tokenization quality we use two datasets: (i) WikiPron (Lee et al., 2020), a multilingual collection of word lists paired with IPA transcriptions extracted from Wiktionary[4] and (ii) FLORES+ (NLLB Team et al., 2024), a dataset of human-translated parallel sentences across more than 200 languages. We use WikiPron for word-level metrics (word fertility and proportion of continued words; see Section 4.4) and FLORES+ for all the remaining metrics, computed at the sentence level. Details can be found in Appendix C.

XNLI and PAWS-X. We fine-tune and evaluate selected models on XNLI (Conneau et al., 2018) and PAWS-X (Yang et al., 2019), two standard multilingual natural language understanding (NLU) benchmarks. These two benchmarks cover 13 and 7 languages from our selection, respectively, as shown in Table 1.

4 Methodology

4.1 Overview

Our experiments compare subword tokenization trained on two input representations: standard orthographic text (**Text**) and its phonemic transcription in the International Phonetic Alphabet (**IPA**). Starting from the same multilingual corpora, we (i) build an IPA version of each dataset with a multilingual grapheme-to-phoneme (G2P) pipeline, (ii) train matched pairs of **Text** vs. **IPA** tokenizers on different configurations, (iii) evaluate intrinsic tokenization quality across languages, and (iv) pre-train GPT-2 models with selected tokenizers and evaluate cross-lingual transfer by fine-tuning on English and testing zero-shot on other languages.

4.2 Grapheme-to-phoneme conversion

To train and evaluate IPA tokenizers and IPA-based language models, we require IPA text at scale. While some resources provide curated IPA transcriptions (e.g., WikiPron (Lee et al., 2020) or Universal Dependencies (UD; Nivre et al., 2020)), these are often limited in size and language coverage. In this work, we utilize Epitran (Mortensen et al., 2018), an open-source rule-based grapheme-to-phoneme (G2P) toolkit designed specifically for multilingual applications. Epitran supports more than 60 languages across a variety of scripts and is being actively extended. It takes standard orthographic text as input, together with the source language and script code (e.g. `tokenization`, `eng-Latn`), and returns its IPA equivalent (e.g. towkənəzejʃən). We applied minor language-specific fixes and extensions, as well as efficiency improvements to Epitran's conversion pipeline, to adapt it best to our needs (see Appendix D for a detailed discussion). For languages that are not supported by Epitran (Greek and Hebrew), we use Phonemizer (Bernard and Titeux, 2021) instead. We discuss G2P conversion quality in Appendix E

4.3 Tokenizer training

To isolate how the input representation affects subword tokenization we vary a set of configurations outlined below. For each configuration we train a paired set of tokenizers: one on **Text** and one on **IPA**. We train all tokenizers with SentencePiece (Kudo and Richardson, 2018) to allow comparisons across subword algorithms under consistent training conditions and because it avoids whitespace-based pre-tokenization, which is important for languages in our study without explicit word boundary markers. We detail hyper-parameter settings and representation-specific adjustments (such as Unicode normalization and character coverage) in Appendix H.1.

Experimental grid. We vary three factors in our experiments: (1) subword tokenization algorithm (BPE, UnigramLM), (2) vocabulary size (40k, 80k, 100k, 200k), and (3) data sampling strategy (`byte-uniform`, `semantic-uniform`, `data-proportional`, `data-smoothed`; defined in Section 4.3). This yields $2 \times 4 \times 4 = 32$ configurations,

[4] https://www.wiktionary.org/

and thus 32 paired Text/IPA tokenizers (64 tokenizers in total).

4.4 Intrinsic evaluation metrics

We assess tokenization quality using ten metrics grouped into four categories—compression, token frequency distribution shape, vocabulary usage, and cross-lingual equity—and refer to Appendix F for formal definitions.

4.4.1 Compression

We report four compression metrics: (i) **Word Fertility (WF)**, the average number of subword tokens per unique word type, where lower values indicate that words are represented with fewer pieces; (ii) **Proportion of Continued Words (PCW)**, the fraction of unique words split into more than one token, where lower values indicate fewer splits; (iii) **Average Token Length (ATL)**, the mean token length measured in input characters of the representation (Text or IPA), where higher values indicate tokens spanning longer segments; and (iv) **Compression Rate (CR)**, the average number of input characters per token at the sentence level, where higher values indicate stronger compression.

4.4.2 Token frequency distribution shape

We characterize the token frequency profile with two metrics: (i) **Rényi Entropy (RE)**, computed from token frequencies on evaluation data and reported for α=1 (Shannon), α=2 (collision), and α=∞ (min-entropy), where higher values indicate more uniform token usage across the vocabulary; and (ii) **Zipf Deviation (ZipfD)**, which measures how closely the empirical rank–frequency curve matches an ideal Zipfian distribution, where lower values indicate closer adherence to Zipf's law (Lotz et al., 2025).

4.4.3 Vocabulary usage

We quantify vocabulary usage with two metrics: (i) **Vocabulary Utilization (VU)**, the share of the learned vocabulary that appears at least once in the evaluation data, where higher values indicate less unused capacity; and (ii) **Type–Token Ratio (TTR)**, the number of distinct token types divided by the total number of produced tokens, where higher values indicate more diverse token usage relative to sequence length.

4.4.4 Cross-lingual equity

We assess cross-lingual equity with two metrics: (i) **Tokenization Parity (TP)**, the average ratio of token counts between English and a target language computed over parallel sentence pairs, where values closer to 1 indicate similar segmentation length (Petrov et al., 2023); and (ii) **Tokenization Fairness Gini (TFG)**, the Gini coefficient over language-level tokenization cost, where cost is defined as token count normalized by input bytes, and lower values indicate more equal tokenization cost across languages (Meister, 2025).

4.5 GPT-2 pre-training and fine-tuning

Tokenizer selection. Since training a language model for all 32 tokenizer settings for both Text and IPA is computationally expensive, we select a small set of tokenizers for GPT-2 pre-training experiments using our intrinsic evaluation. We rank the 32 Text and 32 IPA tokenizers by macro-averaging each intrinsic metric over languages and then aggregating the resulting metric-wise ranks into a single mean-rank score. We select the highest ranked Text and IPA tokenizers, which we refer to as ***Text Opt*** and ***IPA Opt***, respectively. ***Text Opt*** tokenizer is found under the following setting [BPE, 200k, `data-proportional`], while for ***IPA Opt*** it is [UnigramLM, 200k, `byte-uniform`]. Because these optima arise under different settings, the comparison is no longer driven only by the input representation. Therefore, we also select two control tokenizers: Text tokenizer with IPA Opt settings (*Text Subopt*) and IPA tokenizer with Text Opt settings (*IPA Subopt*). We, thus, pre-train four GPT-2 models in total. The ranking procedure and the full ranking results are reported in Appendix G.

Pre-training. We train GPT-2 Small models (Radford et al., 2018, 2019) from scratch on the `data-smoothed` sample of CulturaX for our 18 languages using the four selected tokenizers. Given that all of our tokenizers use a 200k vocabulary, we end up with a total of 240M parameter models (rather than 125M in the original GPT-2 Small, which uses a 50k vocabulary). All hyperparameters are fixed across models and their values are reported in Appendix H.2.

Fine-tuning and evaluation. For both XNLI and PAWS-X, we consider three fine-tuning regimes: (1) monolingual, (2) multilingual, and (3) English-only. In the *monolingual* regime, a pre-trained model is fine-tuned and evaluated independently for each language. In the *multilingual* regime, the model is fine-tuned jointly on data from all languages and evaluated separately on each language.

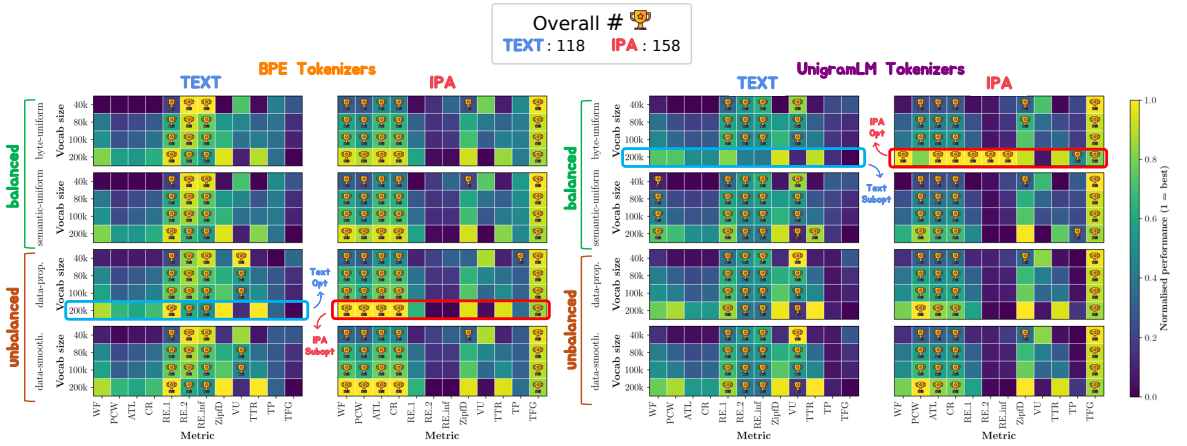


Figure 2: Intrinsic performance across all tokenizer configurations for Text and IPA representations. Each heatmap cell reports a metric value macro-averaged across languages and min–max normalized to $[0, 1]$ (higher is better) for a configuration defined by vocabulary size, subword algorithm, and data-sampling strategy. Cells marked with a trophy indicate a >0.1 advantage over the paired configuration in the other representation (Text vs. IPA). Outlined cells denote the selected *Opt/Subopt* tokenizers for GPT-2 training. **Takeaway:** Across configurations, IPA is consistently stronger on compression (WF, PCW, ATL, CR) and cross-lingual equity (TFG), while Text yields higher token-frequency uniformity as measured by Rényi entropies ($\alpha \in \{1, 2, \infty\}$). Overall, IPA wins more often (158 trophies) than Text (118 trophies).

In the *English-only* regime, the model is fine-tuned exclusively on the English training data and evaluated both on a held-out English split and, in a zero-shot setting, on all remaining languages to measure cross-lingual transfer abilities.

5 Results and Analysis

5.1 Intrinsic tokenization quality across languages and configurations

We begin with two complementary diagnostic views: the configuration-level heatmaps (Fig. 2), which compare intrinsic performance across all tokenizer settings for Text vs. IPA, and a detailed per-language evaluation (Fig. 3) of the optimal tokenizers (with mean statistics for suboptimal ones).

IPA consistently improves tokenization quality. Under their respective optimal settings, IPA improves overall tokenization quality across languages compared to Text (Fig. 3). Across all tested configurations, IPA records more notable wins than Text (158 vs. 118; $\Delta>0.1$ in the normalized metric scale), as shown in Fig. 2, with especially consistent advantages on compression metrics (WF, PCW, ATL, CR) and cross-lingual equity (TFG).Text, by contrast, consistently achieves higher Rényi entropies, indicating a more even spread of token usage across the vocabulary. This pattern is best interpreted in light of IPA's compression gains: higher compression suggests that the learned units are, on average, more word-like, which in turn yields a more Zipfian token distribution, concentrating probability mass more strongly on frequent units while producing a longer tail of rare tokens. Rényi entropy inevitably drops in this scenario, but not as a sign of poorer tokenization, but as a consequence of a more coherent subword structure, which is generally a desirable property. Fig. 5 further summarizes paired effect sizes d_z per metric, aggregated across languages ($d_z > 0 \Rightarrow$ IPA is better than Text; see Appendix N), corroborating that IPA's gains are systematic rather than driven by a single language or setting.

Sensitivity to different configurations. Independent of representation, several configuration trends are stable across metrics: larger vocabularies generally improve compression and yield lower Zipf deviation (best ZipfD typically at 200k), while Rényi entropies show a trade-off for different orders (H_2 and H_∞ strongest at small vocabularies (40k) but H_1 tending to peak at the largest (200k)). Vocabulary usage metrics show a similar pattern: with VU highest at 40k and TTR at 200k vocabularies. Cross-lingual equity prefers smaller vocabularies under the balanced sampling strategies, but shifts this preference towards larger vocabularies under unbalanced mixtures. We provide a more detailed interpretation of these effects in Appendix M. Finally, the optimal Text and IPA tokenizers emerge

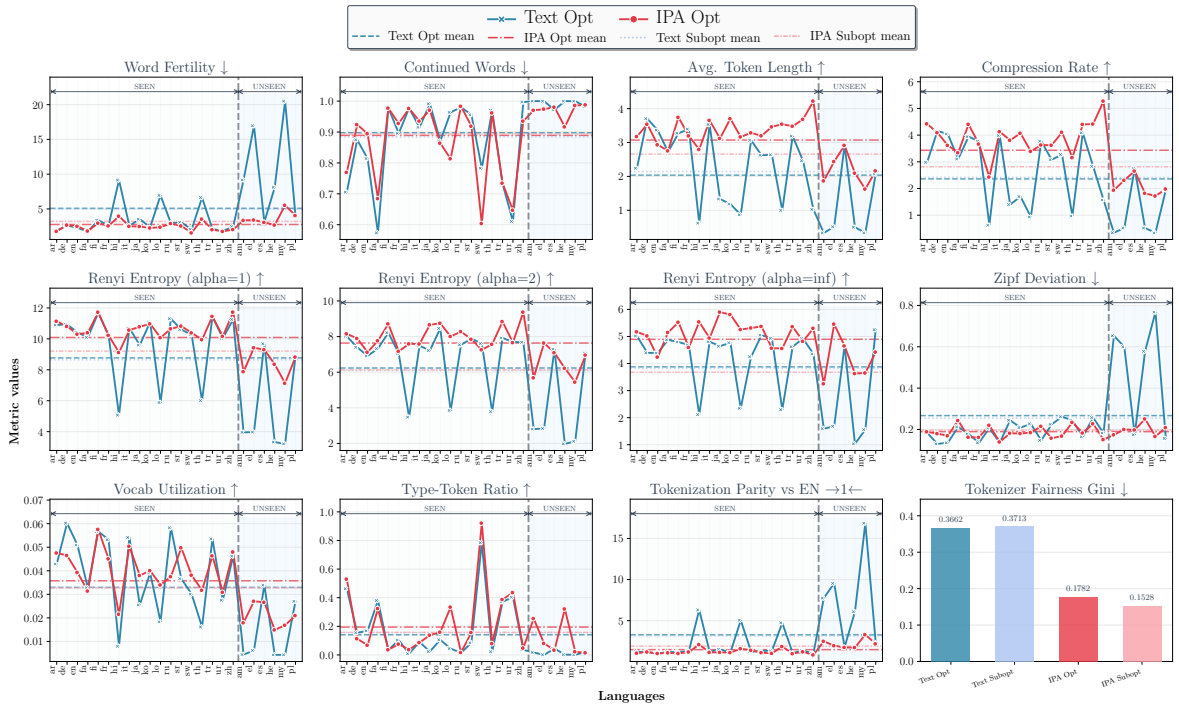


Figure 3: Per-language intrinsic tokenization metrics: **Text** vs. **IPA**. Solid lines show *Text Opt* and *IPA Opt* per language; dashed lines are means. *Subopt* variants shown as mean-only (per-language in Appendix K). Arrows: ↑ = higher is better, ↓ = lower is better, → 1 ← = closer-to-1 is better; TFG is a single global tokenizer metric (not per-language), hence shown as bars. SEEN/UNSEEN mark languages included in vs. absent from tokenizer training data. **Takeaway:** *IPA (Opt)* shows consistently better tokenization quality than *Text (Opt and Subopt)* across metrics, with differences particularly pronounced for unseen languages.

under different configurations, as outlined in Section 4.5 and highlighted in Fig. 2.

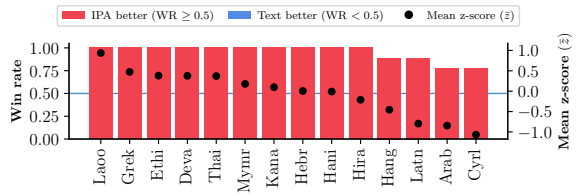
5.1.1 Metric-level analysis

IPA brings largest gains in compression and cross-lingual equity. Figure 5 shows that IPA's largest metric improvements are in compression-related measures (ATL and CR) and cross-lingual equity (TFG), followed by consistent gains in token-frequency distribution (Rényi entropies across different orders). Compression gains indicate that IPA achieves more compact segmentations (fewer tokens for comparable content), while improved equity indicates more uniform tokenization quality across languages. In contrast, improvements are smaller for vocabulary utilization (VU) and continued-word behavior (PCW). The smaller improvements in VU are likely a combined result of our very large vocabularies (200k) and small test sets (WikiPron and FLORES+). With a large vocabulary, many tokens sit in a long tail that never appears in the evaluation sample. Smaller improvements in PCW compared to other compression metrics suggest that while words are overall split into less tokens, the IPA tokenizers still rarely store entire words in their vocabularies.

5.1.2 Language- and script-level analysis

Gains concentrate on non-Latin and unseen scripts. To characterize how different languages are impacted, we use two complementary summaries over intrinsic metrics: *win rate* (WR), the fraction of metrics where IPA outperforms Text for a given language/script, and the *mean z-score* ($\bar{z}$), which summarizes the average standardized magnitude of IPA–Text differences (both described in more detail in Appendix N). The language- and script-level rankings (Fig. 4a, Fig. 4b) show that the largest improvements concentrate on non-Latin scripts, with the strongest gains on unseen scripts (GREK, ETHI, MYMR, HEBR). Languages in these scripts (e.g., EL, AM, MY, HE) achieve WR=1, indicating that IPA improves all intrinsic metrics compared to Text. Substantial gains also appear for complex orthographies and segmentation regimes (e.g., LO, TH, JA (WR=1), and ZH (WR=0.9)), morphologically rich languages (e.g., FI (WR=0.9) and TR (WR=0.72)), as well as low-resource languages (e.g., LO (WR=1) and SW (WR=0.81)). These results uncover a particular fragility of Text tokeniz-

(a) Per-language



(b) Per-script

Figure 4: Ranking **IPA** impact on tokenization quality (a) per-language and (b) per-script. Bars show win rate (WR), the fraction of intrinsic metrics on which *IPA Opt* outperforms *Text Opt*: WR $\in [0, 1]$, with WR$=1/0$ indicating that **IPA**/**Text** wins on all metrics, respectively. **Red** bars mark languages/scripts where **IPA** wins on a majority of metrics (WR>0.5), while **blue** bars mark those where **Text** wins (WR<0.5). Black dots show the **mean z-score** $\bar{z}$ (average standardized IPA–Text delta; $\bar{z}>0$ favors IPA). **Takeaway:** The largest improvements concentrate on unseen scripts (GREK, ETHI, MYMR, HEBR) and on languages with more complex orthographies (e.g., LO, TH, JA), while only a small set of mostly high-resource Latin languages (FR, ES, PL, DE, EN) and one Cyrillic language (RU) show lower overall quality.

ers: when a language or script is underrepresented or completely unseen during tokenizer training, subword algorithms often split words into individual characters or even bytes, leading to fragmented segmentation. In contrast, IPA's shared symbol space across scripts better preserves tokenization quality in these situations.

Only a small set of high-resource Latin languages shows mild degradation. A small subset of languages—EN, DE, FR, ES, PL, RU—shows WR<0.5 (Fig. 4a), meaning Text wins on a majority of intrinsic metrics for these cases. These languages are predominantly high-resource and written in Latin script, which is best represented during tokenizer training, where Text tokenizers already perform well.[5] However, these degradations have relatively small magnitudes ($\bar{z}<0$) compared to the improvements across other languages.

[5]One exception is Russian, which is written in Cyrillic script, but is still one of the better represented languages in our training data.

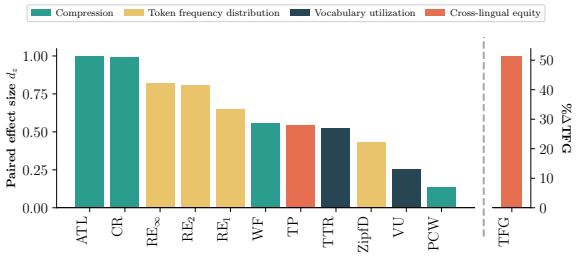
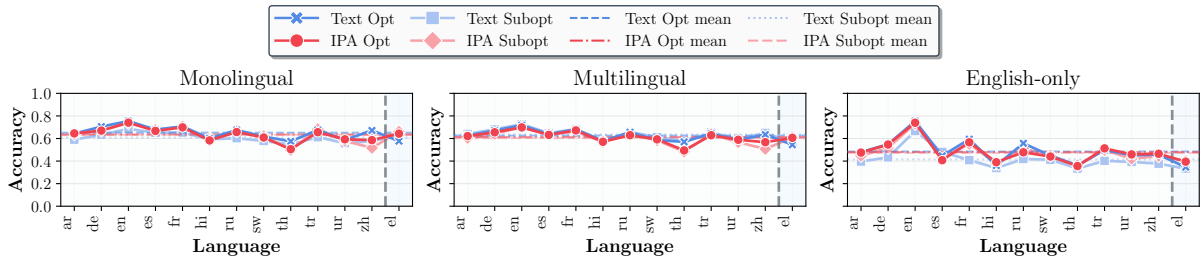


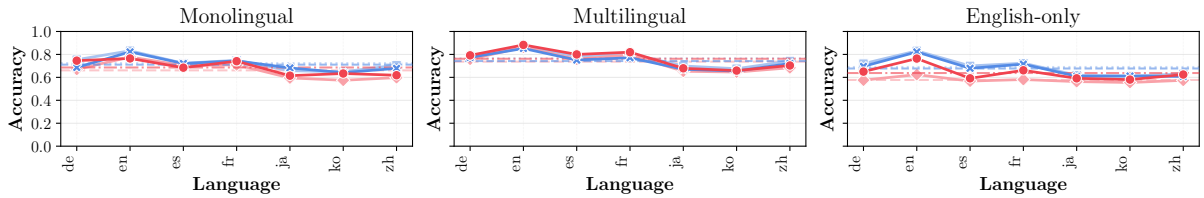
Figure 5: Ranking **IPA** impact on tokenization quality per metric. Bars are grouped and color-coded by metric family (see Sec. 4.4). Left panel reports paired effect size d_z across languages for 11 per-language metrics ($d_z > 0$ favors **IPA**, $d_z < 0$ favors **Text**). TFG is shown separately (right panel) because it is a single global tokenizer statistic rather than a per-language metric; we report its relative change ($\%\Delta\text{TFG}_{\text{Text Opt}-\text{IPA Opt}}$; > 0 favors **IPA**). **Takeaway:** IPA improves average intrinsic quality across all metrics, with the largest gains in compression (ATL, CR) and cross-lingual equity (TFG), suggesting better tokenization efficiency and more uniform cross-lingual behavior.

5.2 Downstream task performance and efficiency

IPA models match downstream accuracy while reducing inference cost. Figure 6 shows per-language accuracies for Text and IPA tokenizers under both *Opt* and *Subopt* configurations, across the three fine-tuning regimes described in Section 4.5. Overall, IPA models achieve mean accuracies comparable to those of Text models on both datasets. On PAWS-X, performance varies modestly across fine-tuning settings: IPA slightly outperforms Text under multilingual fine-tuning, but trails it in the English-only regime. Notably, IPA models outperform Text models on Greek (EL), the only language in either benchmark absent from the tokenizer training data. We also observe a small set of languages (TH, ZH), for which IPA models consistently lag slightly behind Text models; one possible explanation is somewhat lower grapheme-to-phoneme quality for these languages, though a detailed per-language analysis is left for future work. Despite this near-parity in accuracy, the accuracy–compression trade-off in Fig. 7 shows that IPA improves overall compression on both benchmarks. In practice, this means fewer tokens per input and therefore lower inference cost. Consistent with the intrinsic results, the same set of high-resource Latin languages (EN, DE, FR, ES) + RU see mild compression regressions, which are outweighed by more substantial improvements on other languages, resulting in a favorable accuracy–efficiency trade-off: near-par downstream perfor-

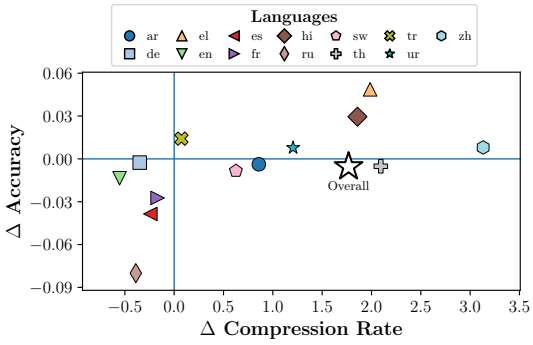


(a) XNLI.

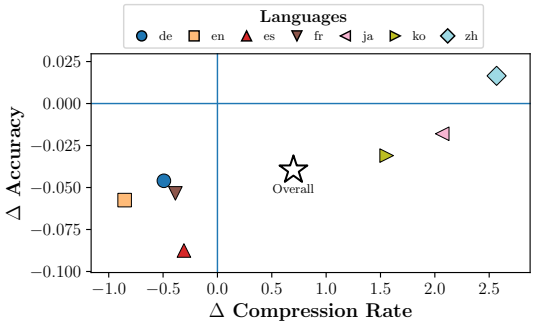


(b) PAWS-X.

Figure 6: Per-language fine-tuning results for (a) XNLI and (b) PAWS-X showing **Text Opt**, **IPA Opt**, **Text Subopt**, and **IPA Subopt** models. We show results for three different fine-tuning strategies (monolingual, multilingual, and English-only). **Takeaway:** On average, **IPA** models are on-par with **Text** on both datasets, with some minor variation across fine-tuning strategies.



(a) XNLI.



(b) PAWS-X.

Figure 7: Accuracy–compression trade-off of **IPA Opt** relative to **Text Opt** on (a) XNLI and (b) PAWS-X. each point is a language with $\Delta\text{Acc} = \text{Acc}_{\text{IPA Opt}} - \text{Acc}_{\text{Text Opt}}$ versus $\Delta\text{CR} = \text{CR}_{\text{IPA Opt}} - \text{CR}_{\text{Text Opt}}$; the large star denotes the macro-average across languages. Points further right indicate better compression and points higher indicate better accuracy. **Takeaway:** **IPA** models improve overall compression, resulting in reduced inference cost.

mance with reduced token counts, translating to lower compute and latency at inference.

6 Conclusion

In this work, we show that using IPA as an input representation results in improved and more equitable multilingual tokenization quality. Across 24 languages and 14 scripts, the optimal IPA tokenizer consistently outperforms the optimal Text tokenizer on our suite of 10 intrinsic metrics, with largest gains in compression and cross-lingual equity, and most pronounced improvements for non-Latin and unseen scripts. Models pretrained on IPA match text-based models on downstream performance (XNLI, PAWS-X), while lowering overall inference cost due to improved compression. These establishes IPA as a promising direction for reducing tokenization-driven disparities in MLMs.

7 Limitations

Dependence on G2P quality. Our approach relies on grapheme-to-phoneme (G2P) tools to obtain IPA transcriptions at scale. In practice, G2P quality is not even across all languages, and rule-based or deterministic tools such as Epitran cannot handle pronunciation variation (e.g., British English vs. American English) or homographs, where identical spellings correspond to different pronunciations (e.g., live as a verb vs. adjective). As multilingual G2P systems improve and become more robust,

IPA-based pipelines should become increasingly practical for large-scale pretraining.

Converting from IPA back to Text. Autoregressive models trained purely on IPA cannot directly generate standard orthographic text. IPA is a lossy encoding, and not always is there an unambiguous mapping back to the original writing system, making straightforward decoding a challenge. Moreover, to the best of our knowledge, there currently exist no publicly available automatic multilingual phoneme-to-grapheme (P2G) conversion tools. Since many real-world uses of generative language models require fluent orthographic output, future work could focus on developing accurate P2G tools or finding ways to incorporate the IPA-to-text mapping mechanism directly into the model architecture.

Generalizability. While we cover a diverse set of languages and vary tokenization and data-sampling configurations to improve generalizability of our findings, our pretrained GPT-2 models (240M parameters) are rather small by current standards. It therefore remains an open question whether the same trade-offs hold at larger scales. In addition, we evaluate downstream transfer on only two tasks and report a single run per model. Stronger evidence would come from broader downstream evaluation and reporting average results over multiple runs to improve robustness.

Representation choices and scope of analysis. In this work, we focus on IPA transcriptions and do not consider romanization as an alternative input representation. Romanization converts text from its original script into a Latin-script representation, offering similar benefits as IPA in terms of reducing script variation and increasing cross-lingual consistency. In addition, our analysis primarily targets the tokenization stage. While we do provide a set of downstream results, we do not provide a detailed per-language analysis. Future work could therefore compare orthographic text, IPA, and romanization more systematically across the stages of pre-training and fine-tuning, and analyze the effects of these different input representations on downstream performance more thoroughly.

8 Acknowledgements

This research was funded/co-funded by the European Union (ERC, CulturAL, 101171968). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah A Smith. 2024. Magnet: Improving the multilingual fairness of language models with adaptive gradient-based tokenization. *Advances in Neural Information Processing Systems*, 37:47790–47814.

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. *arXiv preprint arXiv:2305.13707*.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, and 1 others. 2024. Tokenizer choice for llm training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924.

Catherine Arnett and Benjamin Bergen. 2025. Why do language models perform worse for morphologically complex languages? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6607–6623.

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

Lisa Beinborn and Yuval Pinter. 2023. Analyzing cognitive plausibility of subword tokenization. *arXiv preprint arXiv:2310.13348*.

Mathieu Bernard and Hadrien Titeux. 2021. Phonemizer: Text to phones transcription for multiple languages in python. *Journal of Open Source Software*, 6(68):3958.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media, Sebastopol, CA.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720*.

Iaroslav Chelombitko, Egor Safronov, and Aleksey Komissarov. 2024. Qtok: A comprehensive framework for evaluating multilingual tokenizer quality in large language models. *arXiv preprint arXiv:2410.12989*.

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational 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.

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.

Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0. In *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Jacob Devlin. 2018. Multilingual BERT README. `https://github.com/google-research/bert/blob/master/multilingual.md`. Accessed: 2026-01-05.

Miguel Domingo, Mercedes García-Martínez, Alexandre Helle, Francisco Casacuberta, and Manuel Herranz. 2019. How much does tokenization affect neural machine translation? In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 545–554. Springer.

Darius Feher, Ivan Vulić, and Benjamin Minixhofer. 2025. Retrofitting large language models with dynamic tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29866–29883.

Negar Foroutan, Clara Meister, Debjit Paul, Joel Niklaus, Sina Ahmadi, Antoine Bosselut, and Rico Sennrich. 2025. Parity-aware byte-pair encoding: Improving cross-lingual fairness in tokenization. *arXiv preprint arXiv:2508.04796*.

Robert C Gale, Alexandra C Salem, Gerasimos Fergadiotis, and Steven Bedrick. 2023. Mixed orthographic/phonemic language modeling: Beyond orthographically restricted transformers (bort). In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pages 212–225.

Juan Luis Gastaldi, John Terilla, Luca Malagutti, Brian DuSell, Tim Vieira, and Ryan Cotterell. 2024. The foundations of tokenization: Statistical and computational concerns. *arXiv preprint arXiv:2407.11606*.

Zébulon Goriely, Richard Diehl Martinez, Andrew Caines, Paula Buttery, and Lisa Beinborn. 2024. From babble to words: Pre-training language models on continuous streams of phonemes. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 37–53.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python. Software.

Jue Hou, Anisia Katinskaia, Anh-Duc Vu, and Roman Yangarber. 2023. Effects of sub-word segmentation on performance of transformer language models. *arXiv preprint arXiv:2305.05480*.

International Phonetic Association. 1999. *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press, Cambridge.

Julie Kallini, Dan Jurafsky, Christopher Potts, and Martijn Bartelds. 2025. False Friends are not foes: Investigating vocabulary overlap in multilingual language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21138–21154, Suzhou, China. Association for Computational Linguistics.

Julie Kallini, Shikhar Murty, Christopher D Manning, Christopher Potts, and Róbert Csordás. 2024. Mrt5: Dynamic token merging for efficient byte-level language models. *arXiv preprint arXiv:2410.20771*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, and 1 others. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*.

Jackson L Lee, Lucas FE Ashby, M Elizabeth Garza, Yeonju Lee-Sikka, Sean Miller, Alan Wong, Arya D McCarthy, and Kyle Gorman. 2020. Massively multilingual pronunciation modeling with wikipron. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4223–4228.

Colin Leong and Daniel Whitenack. 2022. Phone-ing it in: Towards flexible multi-modal language model training by phonetic representations of data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5306–5315.

Vladimir I Levenshtein and 1 others. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7871–7880.

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 (Volume 1: Long Papers)*, pages 15059–15076, Bangkok, Thailand. Association for Computational Linguistics.

Jonas F Lotz, António V Lopes, Stephan Peitz, Hendra Setiawan, and Leonardo Emili. 2025. Beyond text compression: Evaluating tokenizers across scales. *arXiv preprint arXiv:2506.03101*.

Jessica M Lundin, Ada Zhang, Nihal Karim, Hamza Louzan, Victor Wei, David Adelani, and Cody Carroll. 2025. The token tax: Systematic bias in multilingual tokenization. *arXiv preprint arXiv:2509.05486*.

Chihaya Matsuhira, Marc A Kastner, Takahiro Komamizu, Takatsugu Hirayama, Keisuke Doman, Yasutomo Kawanishi, and Ichiro Ide. 2023. Ipa-clip: Integrating phonetic priors into vision and language pretraining. *arXiv preprint arXiv:2303.03144*.

Clara Meister. 2025. Tokeval: A tokenizer analysis suite. Software.

Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, and 1 others. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*.

David R Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision g2p for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

David R Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. Panphon: A resource for mapping ipa segments to articulatory feature vectors. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*, pages 3475–3484.

Hoang H Nguyen, Khyati Mahajan, Vikas Yadav, Julian Salazar, Philip S Yu, Masoud Hashemi, and Rishabh Maheshwary. 2024. Prompting with phonemes: Enhancing llms' multilinguality for non-latin script languages. *arXiv preprint arXiv:2411.02398*.

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A Rossi, and Thien Huu Nguyen. 2023. Culturax: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. *arXiv preprint arXiv:2309.09400*.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643*.

Marta R. NLLB Team, 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, and 3 others. 2024. Scaling neural machine translation to 200 languages. *Nature*, 630(8018):841–846.

Abraham Toluwase Owodunni, Orevaoghene Ahia, and Sachin Kumar. 2025. Flexitokens: Flexible tokenization for evolving language models. *arXiv preprint arXiv:2507.12720*.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, and 1 others. 2024. Byte latent transformer: Patches scale better than tokens. *arXiv preprint arXiv:2412.09871*.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *Advances in neural information processing systems*, 36:36963–36990.

Shengju Qian, Yi Zhu, Wenbo Li, Mu Li, and Jiaya Jia. 2022. What makes for good tokenizers in vision transformer? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13011–13023.

Libo Qin, Qiguang Chen, Yuhang Zhou, Zhi Chen, Yinghui Li, Lizi Liao, Min Li, Wanxiang Che, and Philip S Yu. 2025. A survey of multilingual large language models. *Patterns*, 6(1).

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Ms, OpenAI.

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

Bharath Raj, Garvit Suri, Vikrant Dewangan, and Raghav Sonavane. 2024. When every token counts: Optimal segmentation for low-resource language models. *arXiv preprint arXiv:2412.06926*.

Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. 2024. Toward a theory of tokenization in llms. *arXiv preprint arXiv:2404.08335*.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2020. How good is your tokenizer? on the monolingual performance of multilingual language models. *arXiv preprint arXiv:2012.15613*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

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.

Kevin Slagle. 2024. Spacebyte: Towards deleting tokenization from large language modeling. *Advances in Neural Information Processing Systems*, 37:124925–124950.

Jimin Sohn and David R Mortensen. 2025. Cross-lingual ipa contrastive learning for zero-shot ner. *arXiv preprint arXiv:2503.07214*.

Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. 2020. A monolingual approach to contextualized word embeddings for mid-resource languages. *arXiv preprint arXiv:2006.06202*.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache.

Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*.

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. *Transactions of the Association for Computational 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. *Preprint*, arXiv:2010.11934.

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.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. Megabyte: Predicting million-byte sequences with multiscale transformers. *Advances in Neural Information Processing Systems*, 36:78808–78823.

Crystina Zhang, Jing Lu, Vinh Q. Tran, Tal Schuster, Donald Metzler, and Jimmy Lin. 2025. Tomato, tomahto, tomate: Do multilingual language models understand based on subword-level semantic concepts? In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1821–1837, Albuquerque, New Mexico. Association for Computational Linguistics.

Xin Zhang, Dong Zhang, Shimin Li, Yaqian Zhou, and Xipeng Qiu. 2023. Speechtokenizer: Unified speech tokenizer for speech large language models. *arXiv preprint arXiv:2308.16692*.

A International Phonetic Alphabet (IPA)

The International Phonetic Alphabet (IPA) is a system of phonetic notation developed and maintained by the International Phonetic Association to provide a standardized, language-independent representation of the sounds of spoken language. Originally created in the late 19th century, the IPA has undergone multiple refinements over the years, with the most recent official chart standardized in 2020.[6]

The IPA uses a set of dedicated symbols[7] to represent speech sounds, distinguishing four main categories: consonants, vowels, diacritics, and suprasegmentals. Consonant and vowel symbols correspond to specific phonemes and are organized according to articulatory features such as place and manner of articulation for consonants, and tongue height and backness for vowels. Diacritics are used to modify base symbols to indicate finer phonetic

[6] https://www.internationalphoneticassociation.org/content/full-ipa-chart

[7] The exact number varies by source, depending on IPA variants and criteria for what counts as a single character, but typically ranges between 100 and 200 unique symbols.

details such as nasalization, aspiration, or devoicing. Suprasegmentals, such as stress, length, intonation, and tone, are represented with additional symbols and markings to capture prosodic aspects of speech.

Notably, there are two common modes of IPA transcriptions—broad and narrow—which differ in the level of phonetic detail. Broad transcriptions are *phonemic*, in that they only include contrastive sounds needed to distinguish meaning, while narrow transcriptions are *allophonic*, encoding more fine-grained information about phonetic realization and variation. For example, a broad transcription might not differentiate between the sound corresponding to the letter "l" in "leaf" and "pool" (/lif/ and /pul/), while a narrow transcription would ([li:f] and [pu:ɬ])[8,9]. For more information about IPA, one can consult the detailed handbook provided by the International Phonetic Association (1999).

B Tokenization

Tokenization—the process of dividing a sequence of text[10] into smaller, discrete units (i.e. *tokens*)—has been an active area of research in recent years (Hou et al., 2023; Domingo et al., 2019; Rajaraman et al., 2024, *inter alia*). It comes as the first component in the process of training an LLM, and has been shown to have a significant impact on model performance and generalization, especially in multilingual settings (Chelombitko et al., 2024).

Tokenization methods can be systematically classified along two axes: (1) the approach to determining token boundaries—*rule-based* versus *data-driven*; and (2) the granularity of the resulting tokens—*word-level*, *subword-level*, and *character- or byte-level* (Gastaldi et al., 2024). Rule-based tokenization relies on predefined, usually linguistically motivated heuristics, while data-driven approaches learn token boundaries directly from data, typically optimizing for frequency, likelihood, or other corpus-derived statistics. Regarding granularity, in word-level tokenization, each token corresponds to a complete word form. Subword-level tokenization segments text into units smaller than or equal to full words. Character- and byte-level methods treat each character or byte as a separate token, respectively. Table 2 categorizes some notable tokenizers along these axes.

Early NLP systems relied predominantly on rule-based, word-level tokenization. However, this approach encounters significant limitations such as rapid vocabulary growth, frequent out-of-vocabulary (OOV) issues, and necessity for language-specific hand-crafted rules that often require linguistic expertise (Mielke et al., 2021). Data-driven, subword-level tokenizers emerged to address these issues, offering several key advantages. They significantly reduce vocabulary size, improve generalization, and efficiently handle OOV terms by supporting open-vocabulary modeling (Gastaldi et al., 2024). However, subword tokenization is not without drawbacks. Most notably for our discussion, these methods might neglect meaningful linguistic structures (Bostrom and Durrett, 2020; Beinborn and Pinter, 2023) and often unfairly allocate vocabulary space, disproportionately favoring high-resource languages in the training data (Qin et al., 2025; Mielke et al., 2021; Rust et al., 2020). Recently, fully character- and byte-level methods (also collectively known as "tokenization-free" methods) gained attention as alternatives. They entirely avoid vocabulary constraints, providing universal coverage and script independence, alongside robustness to noise (Xue et al., 2022). Yet, these methods also face certain limitations. Firstly, they significantly increase computational cost due to substantially longer input sequences (Kallini et al., 2024). Secondly, by operating on raw bytes, these models inevitably inherit a non-linguistically motivated Unicode[11]-based scheme. In this scheme, every Latin character is encoded in a single byte, whereas characters in many other scripts (e.g. Devanagari, Cyrillic, Han) require two to four bytes per character. Consequently, a Hindi or Chinese sentence of the same length (in terms of characters) as an English one is broken into far more byte-tokens, inflating sequence length and compute and giving those languages a systematic efficiency disadvantage (Mielke et al., 2021). Current research aims to resolve the challenges highlighted above; some notable examples have been highlighted in Section 2.

[8] It would also not differentiate vowel length (indicated by ":"), unless this feature is contrastive (i.e. changes word meaning) in the particular language.

[9] Here we adopt the notation from the official IPA Handbook of enclosing phonemes within forward slashes, i.e. "/…/", and phones within square brackets, i.e. "[…]"

[10] While we focus on text tokenization here, the concept applies more broadly to different types of sequential data e.g., audio stream in speech models (Zhang et al., 2023) or image pixels in computer vision models (Qian et al., 2022).

[11] https://home.unicode.org/

Word-level	Subword-level	Character-/ byte-level
Moses* (Koehn et al., 2007)	Morfessor† (Creutz and Lagus, 2005)	Charformer (Tay et al., 2021)
NLTK* (Bird et al., 2009)	WordPiece† (Schuster and Nakajima, 2012)	ByT5 (Xue et al., 2022)
spaCy* (Honnibal et al., 2020)	BPE† (Sennrich et al., 2016)	CANINE (Clark et al., 2022)
	UnigramLM† (Kudo, 2018)	MegaByte (Yu et al., 2023)
		SpaceByte (Slagle, 2024)
		BLT (Pagnoni et al., 2024)
		MrT5 (Kallini et al., 2024)

Table 2: Representative tokenizers grouped by granularity. Asterisk symbol (*) marks rule-based tokenizers, and dagger symbol (†) data-driven methods. Character-/byte-level column names tokenizer-free models that operate on raw characters/bytes using different approaches.

C Data and languages

For convenience, we provide a mapping between all of the languages considered in this study and their ISO codes in Table 3.

C.1 Mapping of languages and scripts to their ISO codes

Language	ISO 639-1	Script	ISO 15924
Arabic	ar	Arabic	Arab
Amharic	am	Ethiopic	Ethi
Burmese	my	Myanmar	Mymr
Chinese	zh	Han	Hani
English	en	Latin	Latn
Finnish	fi	Latin	Latn
French	fr	Latin	Latn
German	de	Latin	Latn
Greek†	el	Greek	Grek
Hebrew†	he	Hebrew	Hebr
Hindi	hi	Devanagari	Deva
Italian	it	Latin	Latn
Japanese*	ja	Japanese	Jpan
Korean	ko	Hangul	Hang
Lao	lo	Lao	Laoo
Persian	fa	Arabic	Arab
Polish	pl	Latin	Latn
Russian	ru	Cyrillic	Cyrl
Serbian*	sr	Latin/Cyrillic	Latn/Cyrl
Spanish	es	Latin	Latn
Swahili	sw	Latin	Latn
Thai	th	Thai	Thai
Turkish	tr	Latin	Latn
Urdu	ur	Arabic	Arab

Table 3: Languages used in this work. Each language is followed by its two-letter ISO 639-1 code, its script, and the script's four-letter ISO 15924 code. Languages marked with an asterisk (*) use multiple scripts, while languages not supported by Epitran are marked with a dagger symbol (†).

C.2 Data sampling strategies

We visualize the resulting language distributions of the four different data sampling strategies in Fig. 8.

C.3 WikiPron details

Table 4 shows WikiPron dataset details by language on the sample that we use in this work.

C.4 CulturaX post-processing

We apply a lightweight, line-level cleaning step to CulturaX before writing the sampled text to disk. We remove URLs matching `http(s)://…` or `www.…`; remove any remaining tag-like spans of the form `<…>`; remove emoji characters; filter the text to retain only Unicode letters (`L`), numbers (`N`), punctuation (`P`), and standard spaces (`Zs`), deleting all other characters; and finally normalize whitespace by collapsing consecutive whitespace to a single space and stripping leading/trailing spaces. After this pipeline, we discard empty lines and keep non-empty cleaned lines for sampling.

D Epitran usage details

A typical usage involves instantiating an `Epitran` object with a language–script code (e.g., `epitran.Epitran('eng-Latn')` for English) and then calling its `transliterate()` method, which returns the IPA representation. Internally, Epitran relies on language-specific mapping files and optional pre- and post-processing rules. An alternative utility is the `Backoff` class, which allows cascading multiple language–script mappings (for example, to handle mixed-script inputs by falling back from one transliterator to another). However, the `Backoff` class currently does not support the full range of text pre- and post-processing routines applied by the standard `transliterate()`, leading us adopt a different transliteration approach. In this section, we describe this approach, as well as other modifications that we made to this tool.

Bug fixes and mapping extensions. We corrected identified bugs in the Serbian mapping and

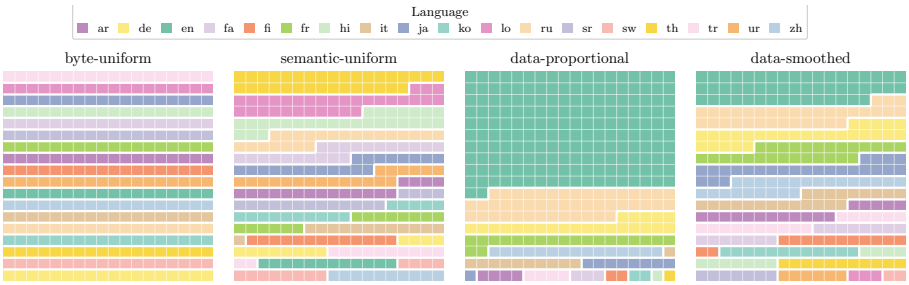


Figure 8: Resulting distribution of languages in the training data using four different sampling strategies. Each square represents $\approx 0.3\%$ of the training data.

added several missing mappings for Urdu, Arabic, Persian, Hindi, Finnish, and Turkish. These additions were based on empirically observed character sequences occurring above 1% frequency in our training data and consulted via well-documented Wikipedia pages of the respective languages. The exact modifications we made per aforementioned language are accessible in our codebase.

Efficient processing for Chinese and Japanese. While most languages were processed by applying `transliterate()` *once* per full document, this approach proved inefficient for Chinese and Japanese due their special treatment by Epitran under the hood, which resulted in a non-linear runtime increase with increased input length (see Figure 9). To address this, we split documents into smaller segments along Chinese/Japanese specific punctuation, ensuring word boundaries were preserved. This chunking preserved performance while maintaining accuracy. In contrast, this segmentation approach results in a notable overhead for other languages (see Figure 10), so it was only applied to Chinese and Japanese.

Sequential mapping for multiple variants. For languages supported by multiple mapping files we employed a specific strategy: we applied the first mapping to the full text, then identified characters still untransliterated (i.e., non-IPA), and selectively applied subsequent mappings only to those residual characters. This procedure ensured each script variant was handled thoroughly, while avoiding unnecessary repeated processing.

Post-conversion cleanup. After IPA conversion, any remaining characters not recognized or mapped by Epitran (primarily external script tokens included in CulturaX data for the particular language, but also, at times, characters not supported by Epitran) were removed. Across languages, this accounted for an average of only 0.3% of tokens per language, a removal rate negligible in volume yet important for ensuring consistency and purity of the phonological representation space.

E G2P conversion quality

We estimated the quality of Epitran grapheme-to-phoneme transcriptions by comparing each predicted IPA form against gold pronunciations from WikiPron using normalized Feature Error Rate (nFER). To motivate this choice, we first briefly relate nFER to the more standard Phoneme Error Rate (PER) and to Feature Error Rate (FER), from which nFER is derived.

Phoneme Error Rate (PER). Let w be a word, let $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ denote its predicted IPA transcription, and let $\mathcal{G}(w) = \{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(K)}\}$ be the set of gold IPA variants in WikiPron. PER is the standard phone-level Levenshtein distance (Levenshtein et al., 1966) between a prediction $\hat{\mathbf{y}}$ and a gold transcription $\mathbf{y}$, normalized by the number of phones in the reference:

$$\mathrm{PER}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\mathrm{ED}_{\mathrm{phone}}(\hat{\mathbf{y}}, \mathbf{y})}{|\mathbf{y}|}, \qquad (1)$$

where $\mathrm{ED}_{\mathrm{phone}}$ uses insertion, deletion, and substitution costs of 1. PER therefore treats all phone substitutions equally, regardless of whether the mismatched phones are phonetically very similar or very different.

Feature Error Rate (FER). FER replaces this binary substitution cost with a feature-based cost derived from PanPhon (Mortensen et al., 2016) articulatory feature vectors. Let $\phi(p) \in \{-1, 0, +1\}^F$ denote the feature vector for phone p, where F is the number of articulatory features. The substitution

Seen Languages (18)					
Language	**ISO**	**Original**	**After Removal**	**Scope**	**Filtered**
Arabic	ar	2 252	1 596	Broad	No
German	de	47 779	42 914	Broad	Yes
English	en	81 576	69 639	Broad	Yes
Persian	fa	34 033	7 357	Narrow	No
Finnish	fi	158 880	156 923	Broad	No
French	fr	80 690	71 229	Broad	Yes
Hindi	hi	24 640	15 867	Broad	Yes
Italian	it	79 865	73 136	Broad	Yes
Japanese*	ja	32 749	29 092	Narrow	Yes
Korean	ko	22 072	21 383	Narrow	Yes
Lao	lo	4 180	2 085	Narrow	No
Russian	ru	411 651	395 845	Narrow	No
Serbian*	sr	46 991	45 553	Broad	Yes
Swahili	sw	110	106	Broad	No
Thai	th	16 689	11 589	Broad	No
Turkish	tr	7 266	6 933	Broad	No
Urdu	ur	4 493	3 637	Broad	No
Chinese	zh	158 873	126 804	Broad	No
Total	—	**1 214 798**	**1 081 688**		
Unseen Languages (6)					
Language	**ISO**	**Original**	**After Removal**	**Scope**	**Filtered**
Amharic	am	378	371	Broad	No
Greek	el	14 825	14 825	Broad	Yes
Spanish	es	99 043	98 787	Broad	Yes
Hebrew	he	1 957	1 945	Broad	No
Burmese	my	6 062	4 486	Broad	Yes
Polish	pl	132 558	115 865	Broad	No
Total	—	**256 608**	**238 057**		

Table 4: WikiPron details by language. "Original" represents the total number of words in the original dataset; "After Removal" shows the number of remaining words after deduplication."Scope" and "Filtered" columns direct to the specific WikiPron file that was used per language (we always selected *broad* and *filtered* versions, unless unavailable). For languages marked with an asterisk (*), we include different scripts (Katakana and Hiragana for Japanese; Latin and Cyrillic for Serbian).

cost between phones p and q is

$$c_{\text{sub}}(p, q) = \frac{1}{F} \sum_{f=1}^{F} \frac{|\phi_f(p) - \phi_f(q)|}{2}. \quad (2)$$

This assigns cost 0 to identical phones, smaller penalties when one feature value is unspecified, and larger penalties when phones differ on many specified features. Insertions and deletions are defined analogously from the feature vector of the inserted or deleted phone, yielding a feature-based edit distance $\text{ED}_{\text{feat}}(\hat{\mathbf{y}}, \mathbf{y})$. FER is then

$$\text{FER}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\text{ED}_{\text{feat}}(\hat{\mathbf{y}}, \mathbf{y})}{|\mathbf{y}|}. \quad (3)$$

Normalized Feature Error Rate (nFER). This metric uses the same feature-based edit distance as FER but normalizes by the length of the longer transcription:

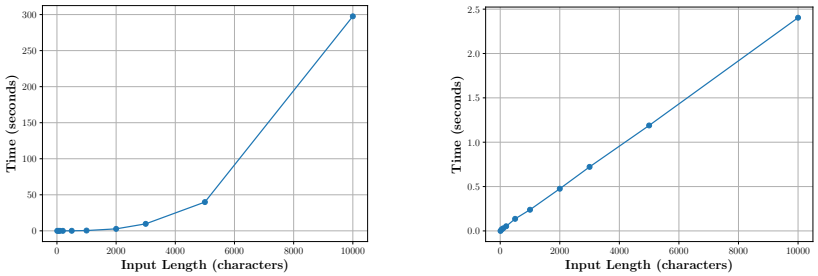
$$\text{nFER}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\text{ED}_{\text{feat}}(\hat{\mathbf{y}}, \mathbf{y})}{\max\big(|\hat{\mathbf{y}}|, |\mathbf{y}|\big)}. \quad (4)$$

This choice makes the score less sensitive to length mismatches between the predicted and gold transcriptions. It is particularly useful here because Epitran and WikiPron sometimes differ slightly in IPA conventions while remaining phonetically close; unlike PER, nFER gives lower penalty to such near-matches by measuring disagreement at the level of articulatory features.

Because WikiPron may provide multiple acceptable pronunciations for a word, we score each prediction against all listed variants and retain the best match:

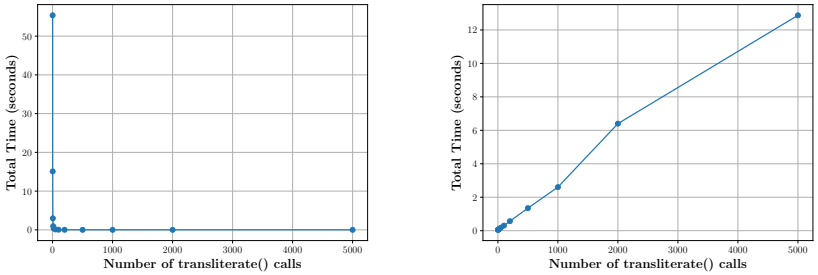
$$\text{nFER}(w) = \min_{\mathbf{y} \in \mathcal{G}(w)} \text{nFER}(\hat{\mathbf{y}}, \mathbf{y}). \quad (5)$$

We then report the mean over all evaluated words



(a) Chinese (b) English

Figure 9: Runtime vs. Input length efficiency trade-off for (a) Chinese and (b) English.



(a) Chinese (b) English

Figure 10: Runtime vs. Number of `transliterate()` calls efficiency trade-off for (a) Chinese and (b) English.

per language:

$$\text{nFER}_{\text{avg}} = \frac{1}{|\mathcal{W}|} \sum_{w \in \mathcal{W}} \text{nFER}(w). \tag{6}$$

The results are shown in Table 5.

Many-to-one collision rates. As we noted in our Limitations section, the mapping between graphemes and phonemes can sometimes be ambiguous. On the one hand, a single orthographic string can be mapped to multiple valid strings in IPA (e.g., English homographs "live" – as an adjective vs. verb). On the other hand, different orthographic strings can be mapped to the same string in IPA (e.g., English homophones "rain" and "reign"). We call these scenarios *one-to-many* and *many-to-one* collisions, respectively. However, in most languages, these phenomena are negligible, as we show in Table 6. While both collision scenarios are non-invertible without context, with the presence of context these can be recovered and learned by the model. Therefore, this is not to be viewed as an inherent downside of IPA, nor a challenge that standard Text models avoid.

F Intrinsic tokenization metrics

F.1 Compression

Let W_ℓ be the set of unique word types for language ℓ, and let $\tau(\cdot)$ be a tokenizer mapping an input string to a token sequence. Denote the number of produced tokens by $n_\tau(x) = |\tau(x)|$ and the character length of a string by $|x|$ (in the tokenizer's input representation).

Word Fertility (WF). Average number of subword tokens per word (lower is better):

$$\text{WF}_\ell = \frac{1}{|W_\ell|} \sum_{w \in W_\ell} n_\tau(w). \tag{7}$$

Proportion of Continued Words (PCW). Fraction of word types split into multiple tokens (lower is better):

$$\text{PCW}_\ell = \frac{1}{|W_\ell|} \sum_{w \in W_\ell} \mathbb{I}[n_\tau(w) > 1]\,. \tag{8}$$

Average Token Length (ATL). Average length of produced tokens, measured in characters of the tokenizer input representation (Text or IPA); higher indicates longer token segments. Let $\mathcal{T}_\ell = \biguplus_{w \in W_\ell} \tau(w)$ be the multiset of all tokens produced for W_ℓ, and let $|t|$ be the character length of token t (excluding any boundary marker, e.g. leading "_"). Then ATL is given as:

$$\text{ATL}_\ell = \frac{1}{|\mathcal{T}_\ell|} \sum_{t \in \mathcal{T}_\ell} |t|. \tag{9}$$

Language	nFER (%)
AM	4.00
AR	24.83
DE	6.77
EN	6.18
ES	0.35
FA	19.15
FI	0.88
FR	7.60
HI	5.22
IT	6.13
JA	1.33
KO	3.24
LO	22.58
MY	9.85
PL	1.14
RU	5.08
SR	0.87
SW	5.60
TH	21.76
TR	1.39
UR	16.55
ZH	26.49

Table 5: Epitran G2P quality measured as average normalized Feature Error Rate (nFER, %) on a WikiPron sample of 2,000 words per language. Lower is better.

Compression Rate (CR). Average number of *raw input characters* per token at the sentence level (higher indicates stronger compression). For a sentence set S_ℓ:

$$\mathrm{CR}_\ell \;=\; \frac{1}{|S_\ell|}\sum_{s\in S_\ell}\frac{|s|}{n_\tau(s)}. \tag{10}$$

F.2 Token frequency distribution shape

Let $c(v)$ be the count of token type v on evaluation data for language ℓ, with total token count $N=\sum_v c(v)$ and empirical probabilities $p(v)=c(v)/N$.

Rényi Entropy (RE). Measures how evenly token usage is distributed across the vocabulary:

$$H_\alpha \;=\; \frac{1}{1-\alpha}\log\sum_v p(v)^\alpha. \tag{11}$$

We report $\alpha=1$ (Shannon entropy), $\alpha=2$ (collision entropy), and $\alpha=\infty$ (min-entropy). Higher values indicate a more uniform distribution.

Zipf Deviation (ZipfD). Following Lotz et al. (2025), we quantify how closely the empirical token rank–frequency distribution follows a Zipfian power law by fitting a linear function to the log–log curve and reporting its mean absolute deviation. Concretely, let tokens be sorted by descending frequency, with rank $r\in\{1,\dots,R\}$ and frequency f_r. Define $x_r=\log r$ and $y_r=\log f_r$, and estimate parameters β_0,β_1 via least squares for $f(x)=\beta_0+\beta_1 x$. Zipf deviation is then

$$\mathrm{ZipfD} \;=\; \frac{1}{R}\sum_{r=1}^{R}\left|\beta_0+\beta_1 x_r-y_r\right|. \tag{12}$$

Lower ZipfD indicates closer agreement with Zipf's law.

Lang.	Collision rate (%)
AM	1.62
AR	2.67
DE	2.99
EL	13.68
EN	10.78
ES	5.39
FA	1.91
FI	0.50
FR	42.98
HE	3.44
HI	5.79
IT	0.75
JA	0.83
KO	1.43
LO	7.47
MY	6.36
PL	3.17
RU	12.01
SR	0.06
SW	0.00
TH	8.00
TR	0.97
UR	2.06
ZH	28.19

Table 6: Empirical many-to-one collision rates (%) per language, estimated on WikiPron. Lower is better.

F.3 Vocabulary usage

Using the same notation above, let V be the learned tokenizer vocabulary with size $|V|$, and let $U=\{v\in V: c(v)>0\}$ be the set of observed token types.

Vocabulary Utilization (VU). Share of the learned vocabulary used at least once on evaluation data (higher is better):

$$\mathrm{VU} \;=\; \frac{|U|}{|V|}. \tag{13}$$

Type–Token Ratio (TTR). Distinct token types relative to the total number of produced tokens (higher indicates more diverse usage):

$$\mathrm{TTR} \;=\; \frac{|U|}{N}. \tag{14}$$

F.4 Cross-lingual equity

Let $S_\ell^{\parallel}$ be a parallel sentence set paired with English, containing pairs $(s_{\mathrm{en}}^{(i)}, s_\ell^{(i)})$.

Tokenization Parity (TP) Following the discussion of Petrov et al. (2023), we compare tokenization length relative to English on parallel sentences:

$$\mathrm{TP}_\ell \;=\; \frac{1}{|S_\ell^{\parallel}|}\sum_i \frac{n_\tau(s_\ell^{(i)})}{n_\tau(s_{\mathrm{en}}^{(i)})}. \tag{15}$$

Values closer to 1 indicate more similar tokenization length relative to English.

Tokenization Fairness Gini (TFG). Following Meister (2025), we measure inequality in tokenization cost across languages using the (equal-weight) Gini coefficient. Let $\mathcal{L} = \{1, \ldots, n\}$ be the set of languages. For each language $\ell \in \mathcal{L}$, define the token cost as

$$c_\ell \;=\; \frac{\sum_{s\in S_\ell} n_\tau(s)}{\sum_{s\in S_\ell} |s|_{\mathrm{bytes}}}, \tag{16}$$

where $n_\tau(s)$ is the number of tokens produced for sentence s and $|s|_{\mathrm{bytes}}$ is its UTF-8 byte length. The mean cost is:

$$\mu \;=\; \frac{1}{n}\sum_{\ell=1}^{n} c_\ell\,. \tag{17}$$

TFG is then:

$$\mathrm{TFG} \;=\; \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|c_i - c_j|}{2n^2\mu}\,. \tag{18}$$

Lower TFG indicates more equal byte-normalized token costs across languages.

G Tokenizer selection

Training autoregressive LMs from scratch for all tokenizer configurations is computationally prohibitive, so we use intrinsic evaluation to select a small set of tokenizers for GPT-2 pre-training. This appendix formalizes the ranking procedure and reports the full ranking heatmaps (Fig. 11).

Set-up. Let $R \in \{\textsc{Text}, \mathrm{IPA}\}$ denote the input representation, and let $\mathcal{C}_R$ be the set of tokenizer configurations evaluated under R (here $|\mathcal{C}_R| = 32$). Each configuration $c \in \mathcal{C}_R$ is defined by the tuple

$$c \;=\; (\text{algorithm}, \text{vocab size}, \text{sampling})\,.$$

Let $\mathcal{L}$ be the set of evaluation languages and let $\mathcal{M}$ be the set of intrinsic metrics. For each language $\ell \in \mathcal{L}$, metric $m \in \mathcal{M}$, and configuration c, we obtain a per-language score $s_m(c, \ell)$ (as defined in Section 4.4).

Macro-averaged metric scores. To obtain a single score per metric and configuration, we macro-average over languages:

$$\bar{s}_m(c) \;=\; \frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}} s_m(c,\ell)\,. \tag{19}$$

In order to make the resulting heatmaps easier to read, we convert all metrics to a consistent higher-is-better orientation by defining:

$$\tilde{s}_m(c) = \begin{cases} \bar{s}_m(c) & \text{if } \uparrow, \\ -\bar{s}_m(c) & \text{if } \downarrow, \\ -|\bar{s}_m(c) - 1| & \text{if } \to 1 \leftarrow, \end{cases} \tag{20}$$

with arrows designating whether the original metric interpretation is higher-is-better (↑), lower-is-better (↓) or closer-to-one-is-better (→ 1 ←).

Per-metric ranks and mean rank aggregation. For each metric m and representation R, we rank configurations by their oriented macro score $\tilde{s}_m(c)$:

$$r_m(c) \;=\; 1 + \sum_{c'\in\mathcal{C}_R} \mathbb{I}\big[\tilde{s}_m(c') > \tilde{s}_m(c)\big]\,; \quad c \in \mathcal{C}_R, \tag{21}$$

where $r_m(c) = 1$ denotes the best configuration for metric m (ties can be handled by average ranks; in practice ties are rare given continuous scores). We aggregate metric-wise ranks into a single scalar score by averaging across metrics:

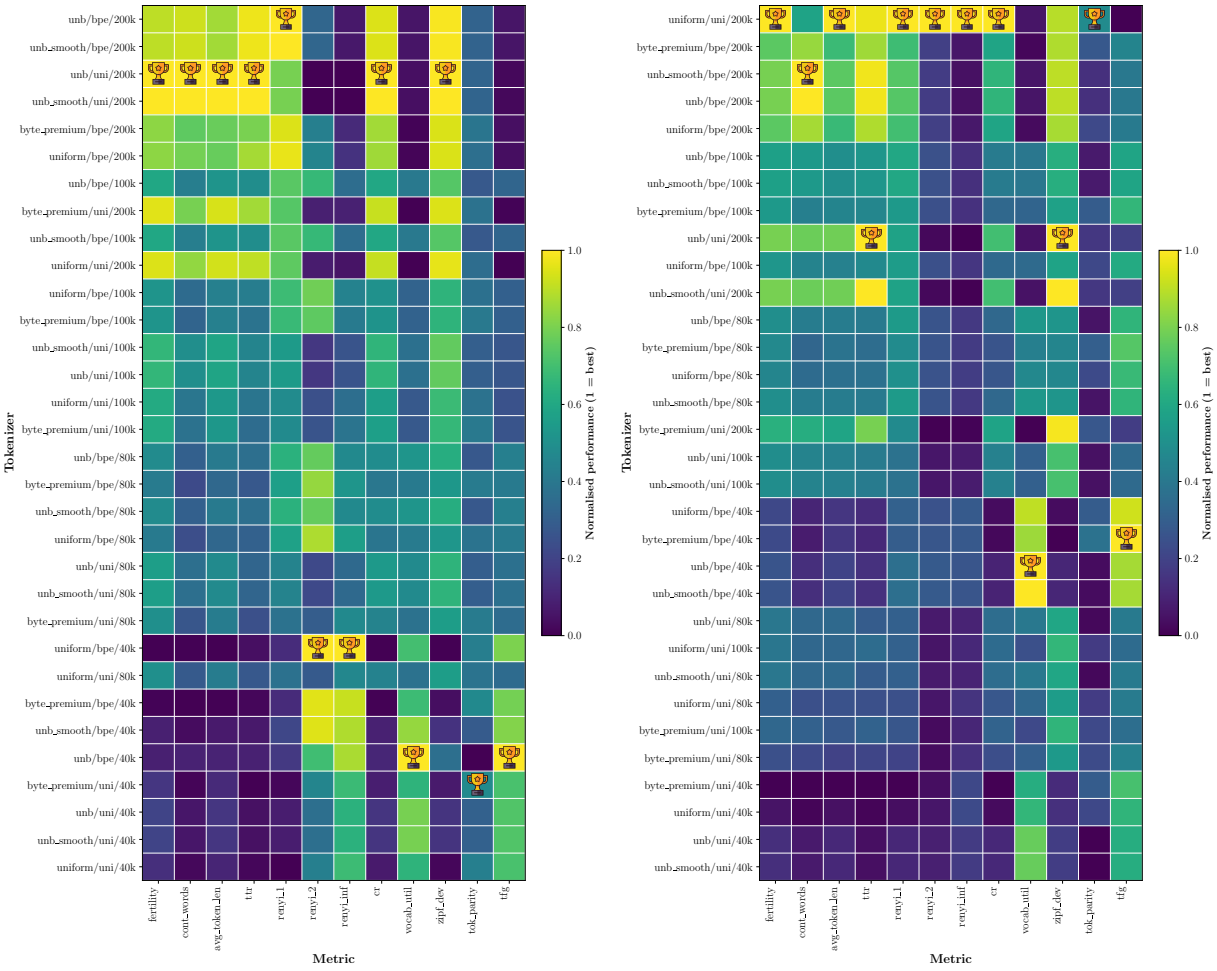
$$\mathrm{MR}(c) \;=\; \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}} r_m(c)\,. \tag{22}$$

Lower $\mathrm{MR}(c)$ indicates a better overall intrinsic ranking. The full rank matrices $\{r_m(c)\}$ and mean-rank scores $\mathrm{MR}(c)$ are visualized in Fig. 11a (**Text**) and Fig. 11b (**IPA**).

Selecting Opt and Subopt tokenizers. For each representation R, we select the intrinsically best tokenizer as

$$c_R^\star \;=\; \arg\min_{c\in\mathcal{C}_R} \mathrm{MR}(c)\,. \tag{23}$$

We refer to the resulting tokenizers as ***Text Opt*** and ***IPA Opt***. In our experiments, ***Text Opt*** corresponds to the configuration $c^\star_{\mathrm{Text}}$ = `(BPE, 200k, data-proportional)`, whereas ***IPA Opt*** corresponds to $c^\star_{\mathrm{IPA}}$ = `(UnigramLM, 200k, byte-uniform)`. Because these optima arise under different hyperparameter settings, they differ along both representation and tokenizer configuration axes. To



(a) Text (b) IPA

Figure 11: Tokenizer selection. We rank Text and IPA tokenizers to select the optimal settings for GPT-2 pre-training. Optimal setting for Text is [BPE, 200k, data-smoothed] and for IPA it is [UnigramLM, 200k, byte-uniform], according to the mean rank across metrics and languages.

control for this, we also select cross-swapped "Subopt" tokenizers: a Text tokenizer trained with the IPA Opt configuration (*Text Subopt*) and an IPA tokenizer trained with the Text Opt configuration (*IPA Subopt*). We then pre-train four GPT-2 models in total: Text Opt, Text Subopt, IPA Opt, and IPA Subopt.

Larger vocabulary sizes. As the optimal tokenizers for both Text and IPA are found on the largest vocabulary size we tested (200k), we consider an even larger vocabulary size of 300k. We find that IPA still outperforms Text on intrinsic metrics with similar trends, though IPA tokenizer's intrinsic quality slightly degrades relative to its 200k version (while TFG improves, suggesting better cross-language balance). Text tokenizer quality is similar between 200k and 300k. However, the previously found optimal settings for both input representations remain optimal even when considering the 300k vocabulary. We provide the results in Table 7.

H Experimental setup

H.1 Tokenizer training hyper-parameters

We provide tokenizer training hyper-parameters in Table 8.

H.2 GPT-2 training and fine-tuning hyper-parameters

We provide GPT-2 pre-training hyper-parameters in Table 9 and fine-tuning hyper-parameters in Table 10.

H.3 Computational footprint

All experiments were run on a single GPU on the Snellius cluster, using either an NVIDIA H100 GPU (for GPT-2 pre-training runs) or an NVIDIA A100 GPU (for all other experiments). The runs

	WF↓	PCW↓	ATL↑	TTR↑	RE_1↑	RE_2↑	RE_∞↑	CR↑	VU↑	ZipfD↓	TP→ 1 ←	TFG↓
Text	4.89	0.92	2.02	0.13	8.81	**6.37**	**3.99**	2.26	0.02	0.29	3.30	0.36
IPA	**3.18**	**0.90**	**2.72**	**0.16**	**9.14**	6.06	3.70	**2.98**	**0.02**	**0.16**	**1.72**	**0.17**

Table 7: Intrinsic tokenization quality at 300k vocabulary size, macro-averaged across languages. Arrows indicate whether lower (↓), higher (↑) or closer-to-1 (→ 1 ←) is better. Best values are boldfaced.

Mode	Parameter	Value
Text	`model_type`	[bpe, unigram]
	`vocab_size`	[40 000, 80 000, 100 000, 200 000]
	`character_coverage`	0.9995
	`split_by_unicode_script`	True
	`byte_fallback`	True
	`normalization_rule_name`	nmt_nfkc
IPA	`model_type`	[bpe, unigram]
	`vocab_size`	[40 000, 80 000, 100 000, 200 000]
	`character_coverage`	1.0
	`split_by_unicode_script`	False
	`byte_fallback`	False
	`normalization_rule_name`	identity

Table 8: Tokenizer training hyper-parameter settings used for Text and IPA representations. We show only the relevant parameters, for all others we used the default SentencePiece values for both modes. Values enclosed in square brackets signify that multiple configurations were considered.

Hyperparameter	Value
Architecture (layers/heads/hidden)	12 / 12 / 768
Context length	2048
Vocabulary size	200k
Training steps	10,000
Batch size (effective)	64
Learning rate	5×10^{-4}
Scheduler	cosine
Warmup ratio	0.03
Weight decay	0.1
Precision	bfloat16
Attention implementation	SDPA
Random seed	42

Table 9: GPT-2 pretraining hyperparameters, shared across all Text/IPA and Opt/Subopt conditions.

Hyperparameter	XNLI	PAWS-X
Number of labels	3	2
Training steps	6128	3860
Batch size (effective)	256	256
Learning rate	1×10^{-4}	3×10^{-5}
Scheduler	cosine	cosine
Warmup ratio	0.06	0.06
Weight decay	0.0	0.0
Max grad norm	1.0	1.0
Precision	bfloat16	bfloat16
Model selection	validation loss	validation loss

Table 10: Fine-tuning hyperparameters for XNLI and PAWS-X, shared across all Text/IPA and Opt/Subopt conditions within each task.

conducted on a single A100 GPU used 85.74 GPU-hours, corresponding to an estimated 21.43 kWh of electricity consumption and 5.61 kg CO2eq. The runs conducted on a single H100 GPU used 27.93 GPU-hours, corresponding to an estimated 19.55 kWh and 5.12 kg CO2eq. In total, our experiments used 113.67 GPU hours, 40.98 kWh of electricity, and produced 10.73 kg CO2eq. Estimations were conducted using the MachineLearning Impact calculator presented in Lacoste et al. (2019).

I General benefits of IPA as an input representation

In the introduction section, we have outlined several beneficial properties offered by the IPA. Here we show what implications those properties have in practice.

IPA improves cross-lingual sharing of character inventories. In Figure 12 we show the percentage of shared characters across each pair of our 18 languages on the CulturaX sample we used for pre-training on the original text and on the converted IPA text. In Text heatmap, we see biggest sharing between languages that use the same script, but that percentage drops for language pairs with different scripts. We see that languages with complex orthographies (Chinese, Japanese, and Korean) have almost zero overlap with any other language. Other languages do show some percentage of overlap even though they differ in scripts, partly because many languages in our sample do contain

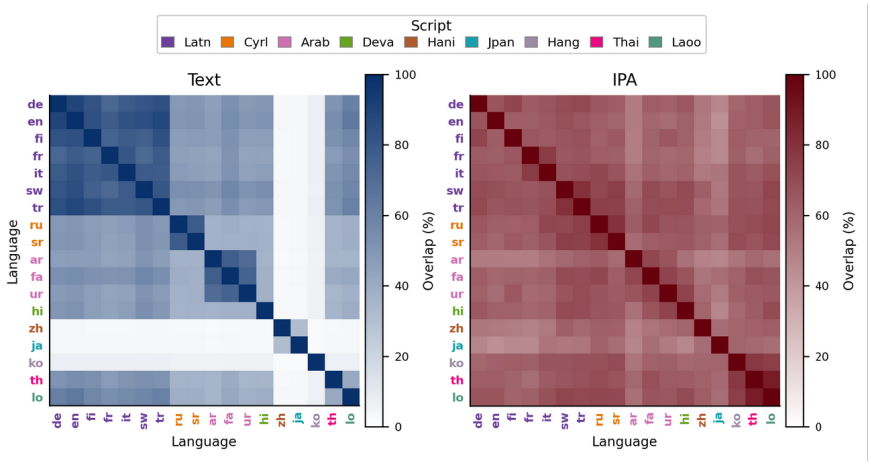


Figure 12: Pairwise overlap of character inventories (%) between the 18 pre-training languages on the CulturaX `byte-uniform` sample, computed on the original Text input (left) and its IPA transcription (right). Each cell reports the percentage of unique characters shared by a language pair (diagonal = 100%). In Text, overlap is largely confined to languages that share a script, with near-zero sharing for languages with distinct orthographies (e.g., ZH, JA, KO); in IPA, overlap becomes higher across languages and more uniform across scripts due to the shared phonemic symbol inventory.

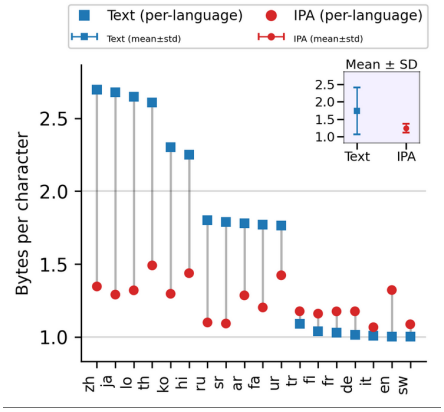


Figure 13: Average bytes-per-character (BPC) by language on the CulturaX `byte-uniform` sample, for the original Text input and its IPA transcription. Text exhibits large cross-lingual disparities, with higher BPC for languages whose scripts occupy higher Unicode ranges (e.g., ZH, JA, LO, TH), while Latin-script languages cluster near 1 BPC. IPA concentrates most symbols in low-byte code points, reducing mean BPC and making storage cost more uniform across languages; the inset (top-right) summarizes the mean ± standard deviation across languages.

inputs from other languages, most notably English. However, the IPA heatmap reveals a much more evenly distributed and overall larger character overlap across languages. This is a direct consequence of IPA's shared input representation for all of the languages. Overall, this increased sharing is beneficial for multilingual tokenizers, as cross-lingual token overlap has been shown to improve model performance (Zhang et al., 2025; Kallini et al., 2025).

IPA reduces storage overhead across languages on average. Figure 13 compares the average number of bytes needed to encode a single character per language for Text vs IPA, estimated from our pre-training CulturaX sample. Standard text representation shows notable disparities: languages with complex orthographies (Chinese, Japanese, Lao, and Thai) require much larger amount (> 2.5) of bytes-per-character (BPC), while languages using Latin script are all very close to 1 byte-per-character. This difference is due to the position of different characters in the Unicode range. The symbols used by the IPA mostly use ASCII characters (1 BPC), together with a set of special IPA symbols (2 BPC). Some additional, but less frequent IPA markers require 3 BPC. This fact, together with IPA's shared language representation space, reduces the average BPC across our set of languages, requiring slightly larger numbers for Latin-script languages than Text, but much lower BPC for all other ones. In addition, the required storage memory is much more balanced across languages.

J IPA for byte-level tokenization

The benefits of IPA as an input representation highlighted in Appendix I also translate to byte-level tokenization-free approaches. In Table 11 we empirically show that, when considering individual bytes as tokens, working with IPA input still results in shorter average sequence lengths compared to using Text inputs. It also reduces the standard devia-

Lang.	Text	IPA
AM	220	**198**
AR	204	**188**
DE	**151**	186
EL	278	**153**
EN	**128**	161
ES	**155**	160
FA	218	**146**
FI	**141**	160
FR	**158**	169
HE	177	**118**
HI	328	**219**
IT	**152**	162
JA	162	**155**
KO	**153**	173
LO	351	**207**
MY	452	**251**
PL	**144**	174
RU	255	**183**
SR	231	**141**
SW	**134**	146
TH	354	**218**
TR	**143**	164
UR	**225**	246
ZH	**118**	167
mean $\pm$ *std.*	210 ± 87	$\mathbf{177 \pm 32}$

Table 11: Average number of tokens per sentence on FLORES+ under byte-level tokenization for Text and IPA input. Boldface marks the lower value within each row (lower values are better). Macro-level mean$\pm$std. are shown at the bottom.

tion, making the sequence length distribution more stable across languages. Therefore, we believe IPA has a potential to be applied on top of the existing byte-level methods to further increase cross-script robustness.

K Per-language results for suboptimal tokenizers

We provide a detailed per-language intrinsic evaluation of the suboptimal tokenizers in Fig. 14.

L Detailed per-script daignostic

We provide a detailed per-script intrinsic evaluation for both the optimal (Fig. 15) and suboptimal tokenizers (Fig. 16).

M Interpretation of tokenizer configuration effects

Compression metrics improve with vocab size. This is consistent across algorithms, sampling strategies, and modes (text/ipa). Consequently, vocab size of 200k performs best in terms of compression (WF, PCW, ATL, CR). This is expected, as larger vocabs will have more space for storing tokens, which means fewer tokens needed to represent the same input.

Renyi H_2 and H_∞ best at smallest vocab; H_1 best at 200k. This pattern is consistent with the different sensitivities of R'enyi orders: H_1 (Shannon entropy) reflects the full distribution and increases when the vocabulary grows to include many rare, low-probability types (e.g., longer tokens approaching full words). In contrast, H_2 and especially H_∞ place increasing weight on the head of the distribution and thus tend to favor settings where probability mass is less concentrated in a small set of very frequent tokens—often achieved by smaller vocabularies that encourage greater sharing of subword units. Because the entropy scale depends strongly on vocabulary size, we treat these trends primarily as descriptive of configuration effects and focus comparisons on the Text vs. IPA axis under matched vocabulary sizes. One exception is `[IPA, UnigramLM, byte-uniform]`, where all three entropies are maximized at 200k; we leave a more detailed analysis of this case to future work.

Zipf Deviation decreases with vocabulary size. Across all settings, Zipf Deviation (ZipfD) is lowest at the largest vocabulary (200k). In Fig. 2, ZipfD appears as higher-is-better because we invert and normalize metrics for readability; in the original definition, lower ZipfD indicates that the empirical rank–frequency curve is closer to an ideal Zipfian trend. The observed monotonic improvement with vocabulary size is intuitive: as vocabularies grow, token inventories increasingly contain longer units (often close to whole-word tokens), and the resulting token frequency distribution more closely mirrors the underlying word-frequency distribution, which is approximately Zipfian in natural language. At the same time, ZipfD is strongly affected by vocabulary granularity, so we interpret the vocabulary-size trend as a configuration effect and emphasize comparisons along the Text vs. IPA axis under matched vocabulary sizes.

Vocabulary utilization always best for 40k, while TTR always best for 200k. Although both metrics quantify vocabulary usage, they capture different aspects of the token distribution. VU measures the fraction of the learned vocabulary that is observed at least once on evaluation data; smaller vocabularies encourage heavier reuse of a limited set of subword units, making it more likely that most entries are encountered. In contrast, TTR measures

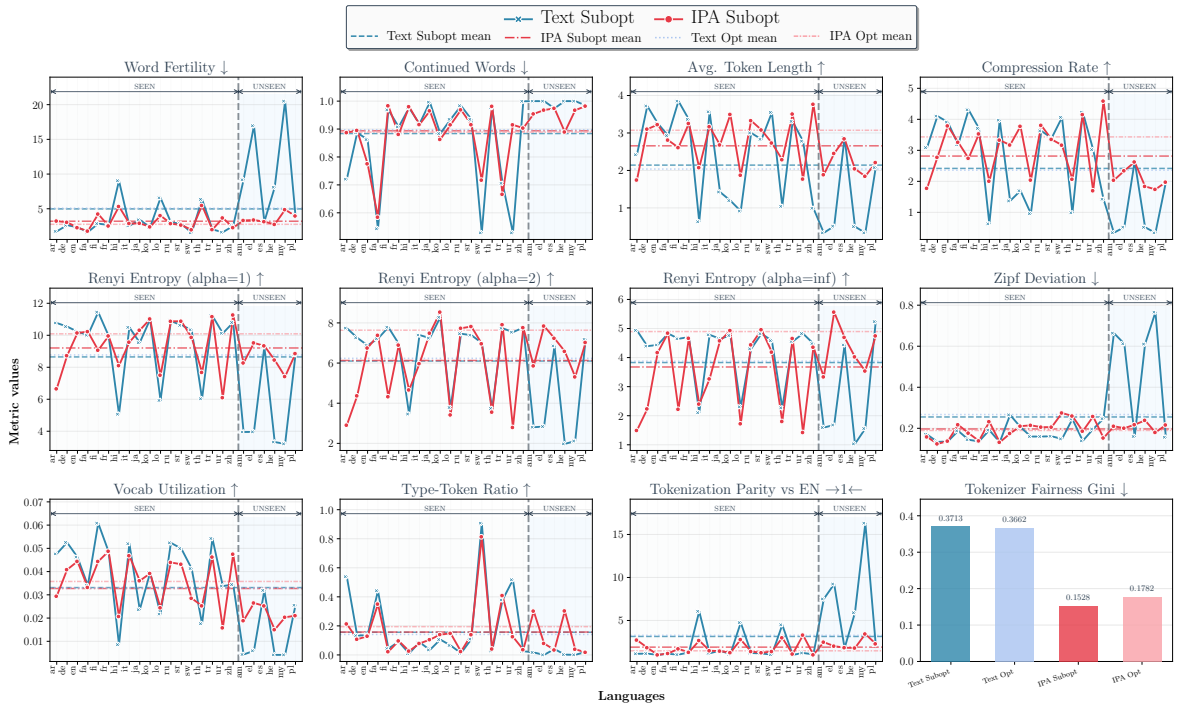


Figure 14: Detailed per-language performance of the suboptimal tokenizers.

the number of distinct token types relative to the total number of produced tokens; larger vocabularies introduce more specific (and often rarer) token types, increasing the diversity of observed types even if many are used only infrequently.

N Additional statistics: win-rate, paired effect size, and mean z-score

We summarize intrinsic differences between IPA and TEXT using (i) a per-language win-rate, (ii) a paired effect size d_z per metric, and (iii) an overall per-language score based on mean z-scored improvements. All computations are paired on the intersection of languages (or scripts) available for both representations. We exclude TFG from per-language/script computations because it is a global metric.

Direction-corrected improvement. For metric m and language/script i, let $x_{m,i}^{\text{IPA}}$ and $x_{m,i}^{\text{TEXT}}$ be the corresponding metric values. We convert each metric to an "IPA-better" improvement value $\Delta_{m,i}$:

$$\Delta_{m,i} = \begin{cases} x_{m,i}^{\text{IPA}} - x_{m,i}^{\text{TEXT}} & (\uparrow) \\ x_{m,i}^{\text{TEXT}} - x_{m,i}^{\text{IPA}} & (\downarrow) \\ |x_{m,i}^{\text{TEXT}} - 1| - |x_{m,i}^{\text{IPA}} - 1| & (\rightarrow 1 \leftarrow) \end{cases} \tag{24}$$

Here $(\uparrow)$ denotes metrics where larger values are better, $(\downarrow)$ denotes metrics where smaller values are better, and $(\rightarrow 1 \leftarrow)$ denotes metrics where values closer to 1 are better. Thus, $\Delta_{m,i} > 0$ means IPA is better than TEXT on metric m for item i.

Win-rate (per language/script). Let $\mathcal{M}_i$ be the set of metrics available for item i. The win-rate is the fraction of metrics improved by IPA:

$$\text{WinRate}(i) = \frac{1}{|\mathcal{M}_i|} \sum_{m \in \mathcal{M}_i} \mathbb{I}[\Delta_{m,i} > 0]\,. \tag{25}$$

Paired effect size d_z (per metric). For each metric m, we compute the mean and standard deviation of $\Delta_{m,i}$ across items $i \in \mathcal{I}_m$ (the items with non-missing paired values):

$$\begin{aligned} \bar{\Delta}_m &= \frac{1}{|\mathcal{I}_m|} \sum_{i \in \mathcal{I}_m} \Delta_{m,i}, \\ s_m &= \text{std}(\{\Delta_{m,i}\}_{i \in \mathcal{I}_m})\,. \end{aligned} \tag{26}$$

We then report Cohen's paired standardized mean difference:

$$d_z(m) = \frac{\bar{\Delta}_m}{s_m + \varepsilon}. \tag{27}$$

where ε is a small constant for numerical stability. Positive d_z indicates that IPA tends to outperform TEXT on metric m.

Mean z-score (overall per language/script). To combine metrics with different scales, we z-score

Figure 15: Per-script intrinsic tokenization metrics: **Text Opt** vs. **IPA Opt**. Solid lines show *Text Opt* and *IPA Opt* per language; dashed lines are means. Arrows: ↑ = higher is better, ↓ = lower is better, → 1 ← = closer-to-1 is better; TFG is a single global tokenizer metric (not per-language), hence shown as bars. Zipf Deviation and Tokenization Parity are per-language only, so they are omitted here. SEEN/UNSEEN mark languages included in vs. absent from tokenizer training data. **Takeaway:** *IPA* improves tokenization quality for non-Latin scripts, while maintaning the same quality on Latin script languages as *Text* tokenizers. Improvements are most drastic for scripts unseen during tokenizer training.

improvements within each metric across items:

$$z_{m,i} = \frac{\Delta_{m,i} - \mu_m}{\sigma_m}, \tag{28}$$

where μ_m and σ_m are the mean and standard deviation of $\Delta_{m,i}$ over $i \in \mathcal{I}_m$ (if $\sigma_m = 0$, we set $z_{m,i} = 0$). The overall score for item i is the mean z across its available metrics:

$$\text{MeanZ}(i) = \frac{1}{|\mathcal{M}_i|} \sum_{m \in \mathcal{M}_i} z_{m,i}. \tag{29}$$

Higher MeanZ(i) indicates that IPA improves more metrics, and by a larger margin, relative to TEXT.

O Sensitivity to noise injection

As an additional experiment, we test robustness of our **IPA Opt** tokenizer to noise injection and compare it to that of the **Text Opt** tokenizer. Specifically, we test sensitivity to random character deletions with three different rates (5%, 10%, and 15%). We provide results in Table 12.

Figure 16: Per-script intrinsic tokenization metrics: Text Subopt vs. IPA Subopt. Solid lines show *Text Subopt* and *IPA Subopt* per language; dashed lines are means. Arrows: ↑ = higher is better, ↓ = lower is better, → 1 ← = closer-to-1 is better; TFG is a single global tokenizer metric (not per-language), hence shown as bars. Zipf Deviation and Tokenization Parity are per-language only, so they are omitted here. SEEN/UNSEEN mark languages included in vs. absent from tokenizer training data.

Mode	Del. rate	WF↓	PCW↓	ATL↑	TTR↑	RE_1↑	RE_2↑	RE_∞↑	CR↑	VU↑	ZipfD↓	TP→ 1 ←
Text Opt	5%	5.89	0.90	1.99	0.13	9.39	6.70	4.17	2.47	0.04	0.24	2.67
	10%	6.02	0.90	1.90	0.12	9.28	6.58	4.08	2.41	0.04	0.25	2.74
	15%	6.15	0.90	1.83	0.12	9.16	6.46	3.99	2.35	0.03	0.26	2.82
IPA Opt	5%	2.84	0.90	2.85	0.17	10.20	7.75	4.97	3.52	0.04	0.19	1.42
	10%	2.85	0.91	2.66	0.16	10.08	7.63	4.88	3.46	0.04	0.20	1.47
	15%	2.83	0.91	2.52	0.16	9.95	7.50	4.79	3.39	0.03	0.21	1.53

Table 12: Sensitivity to noise injection (random character deletions) of Text Opt and IPA Opt tokenizers.