MoTo: Mixture of Tokenizers Towards Fair Multilingual Language Modeling

Gül Sena Altıntaş **Colin Raffel**
University of Toronto, Vector Institute
Canada
{gsaltintas,craffel}@cs.toronto.edu

Abstract

Large language models rely on a single tokenizer that is chosen at training time and fixed thereafter. Most tokenizers are trained on English-dominant corpora, making them under-serve morphologically rich and non-Latin-script languages, where they can produce fragmented representations, longer sequence lengths, and reduced information density during training. We propose *Mixture of Tokenizers* (MoTo), a modular tokenization framework that trains dedicated per-language BPE tokenizers and composes them into a unified superset vocabulary, where tokens shared across languages are deduplicated into a common ID space. Each tokenizer maintains its own normalization policy, pre-tokenization rules, and vocabulary budget. Despite allocating significantly fewer entries to English, MoTo retains English performance while delivering consistent gains across 18 out of 21 typologically diverse languages spanning six scripts, with improvements most pronounced on morphologically complex and non-Latin-script languages that are systematically underserved by English-dominant tokenizer design. Our results suggest that modular subword tokenization is a practical and extensible alternative to monolithic tokenization for multilingual parity.

🤗 **Models & Tokenizers:** r-three-moto
Code: r-three/moto, r-three/lingua
Project page: gsaltintas.github.io/moto
Explore tokenizers: tokenizers.gsaltintas.com

1 Introduction

Modern language models (LMs) are trained and deployed across a wide range of languages and scripts, yet rely on a single static tokenizer that is fixed early on. This is not an incidental limitation: tokenization is the first and least revisited design decision in language model training [Alqahtani et al., 2026], and its consequences propagate forward into downstream representations. A base model is trained on general-purpose web corpora (e.g., Penedo et al. [2024], Grattafiori et al. [2024]), and its fine-tuned, instruction-tuned, and RLHF-aligned variants inherit the same tokenizer. However, many widespread tokenizers are trained primarily on English-dominant natural text, resulting in suboptimal performance when deployed across non-English languages [Rust et al., 2021, Ali et al., 2024].

A tokenizer consists of a vocabulary of subword units associated with learned embedding vectors and a segmentation algorithm that maps raw text into sequences over that vocabulary, both of which are typically learned together based on a target vocabulary size and a choice of learning algorithm. The vocabularies learned by two dominant learning algorithms, byte-pair encoding (BPE) [Sennrich et al., 2016, Gage, 1994] and UnigramLM [Kudo and Richardson, 2018], are both primarily determined by the choice of training corpus and target vocabulary size, alongside discrete design decisions such as normalization rules and pre-tokenization patterns.

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

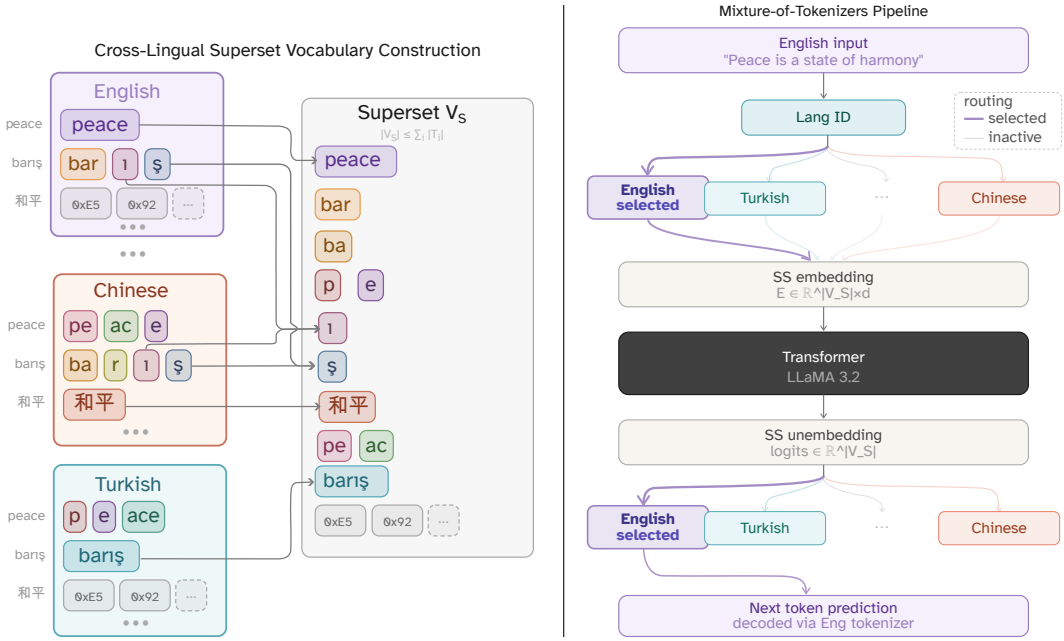

Figure 1: *Left: Superset Vocabulary construction.* Language-specific subword tokenizers are trained on their corresponding corpus. Words native to that language tend to be represented with fewer tokens, while non-native words fall back to coarser subwords or byte-level tokens. The superset vocabulary $\mathcal{V}_S$ is constructed by taking the union of all tokenizer vocabularies, assigning shared tokens a single global index, with $|\mathcal{V}_S| \leq \sum_i |T_i|$. *Right: Mixture-of-Tokenizers (MoTo) pipeline.* A tokenizer is selected based on the input language ID. The input is encoded into superset indices, processed by the transformer, and decoded back with the tokens from the superset, additionally it can utilize rules from the input tokenizer.

However, designing a good tokenizer is far from straightforward. Byte-level compression, measured as the average number of tokens required to encode a given amount of text, is the dominant proxy [Goldman et al., 2024], as it directly affects how much information the model can consume during training and its inference efficiency, but it is an insufficient measure of quality [Schmidt et al., 2024]. Early tokenizers, largely designed for single languages or domains, incorporated strong domain-specific inductive biases through pre-processing choices that improved compression and simplified vocabulary. Both BERT [Devlin et al., 2019] and T5 [Raffel et al., 2020] collapse multiple whitespace characters into one; benign for natural language but destructive for Python, where indentation is syntactically significant. Each of these decisions was locally motivated: it improved compression or reduced sparsity within the target domain. But as models, and therefore tokenizers, expanded to cover more languages and scripts, these assumptions became constraints. The field has since converged toward minimal pre-processing interventions, with Llama-3.2 [Grattafiori et al., 2024], Qwen-3 [Yang et al., 2025], Tekken [AI, 2024], Comma [Kandpal et al., 2026], and Aya [Üstün et al., 2024] all adopting variants of the GPT-2 pre-tokenization regex [Radford et al., 2019, OpenAI et al., 2024], likely motivated by the following observation:

O.1 *Preprocessing choices that improve efficiency for a target domain encode inductive biases that might not transfer across languages and scripts.*

Even so, minimizing pre-processing and treating tokenization as an end-to-end problem of finding co-occurring byte sequences does not resolve the issue that learned merges are driven by frequency, favoring languages that are better represented in the training corpus, leaving others with less vocabulary coverage. A related but distinct concern is representability: to guarantee that *any* input can be represented regardless of script or training coverage, most tokenizers adopt *byte fallback* so that any character unseen during training can still be encoded in terms of its UTF-8 byte representation.

Script-level asymmetries compound these disparities: languages without whitespace delimiters, like Chinese and Japanese, require explicit pre-tokenization decisions. Such differences in writing systems

can lead to substantial differences in the number of tokens needed to encode equivalent content, a phenomenon known as the token premium [Petrov et al., 2023]. Moreover, the byte premium [Arnett et al., 2024] of non-Latin scripts can translate into additional vocabulary overhead, as encoding a multi-byte character may require intermediate BPE merges, resulting in *scaffold tokens* that serve primarily as building blocks for larger tokens [Lian et al., 2024]. Differences in byte-level representation can further affect vocabulary allocation, as frequent byte sequences and intermediate merges may occupy vocabulary slots without providing equally useful units across languages [Arnett et al., 2024, Lian et al., 2024]. Some models address these imbalances by augmenting their vocabularies with frequent words from particular languages. For instance, Qwen [Yang et al., 2025] and DeepSeek [DeepSeek-AI et al., 2025] augment their vocabularies with frequent Chinese words, but such adjustments are ad hoc and do not generalize across languages or domains.

More fundamentally, the learned vocabulary itself is rarely the object of deliberate design: it is treated as a byproduct of corpus frequency and vocabulary size, investigated only after the fact if at all. This has concrete consequences for multilingual fairness. For instance: despite its large vocabulary, Gemma-2 [Team et al., 2024] dedicates six separate entries to surface-level variations of the word `hello` (`hello`, `_hello`, `Hello`, `_Hello`, `HELLO`, `_HELLO`), resulting in embedding table entries allocated to English orthographic redundancy that could otherwise serve underrepresented languages. These limitations motivate a fundamental reexamination of how tokenizer vocabularies are constructed:

O.2 *A single jointly trained tokenizer cannot simultaneously serve all languages and domains well and a large vocabulary size does not guarantee multilingual or multi-domain coverage.*

Recent work has sought to address these limitations through linguistically motivated relaxations [Liu et al., 2025, Schmidt et al., 2025], further refinement of vocabulary [Chizhov et al., 2024, Bauwens and Delobelle, 2024] or by transferring and adapting existing tokenizers to new languages post-hoc [Tran, 2020, Minixhofer et al., 2022, Gee et al., 2022, Dobler and de Melo, 2023, Liu et al., 2024, Goddard and Neto, 2025, Minixhofer et al., 2024]. In this paper, we question the fundamental assumption that a single jointly trained tokenizer should serve all languages and instead propose `MoTo`: **Mixture of Tokenizers** as a modular-first design that treats the tokenizer as a composable artifact rather than a monolithic decision. Concretely, we make the following contributions:

- **Modular superset tokenization.** We train small, dedicated BPE tokenizers per language across 21 languages with language-specific design choices, and compose them into a unified superset vocabulary without modifying the underlying model architecture (§ 2.1 and Fig. 1).
- **Controlled multilingual benchmark.** We introduce a synthetic multilingual addition task, and show that `MoTo` with a superset of 2K tokens outperforms all variants while linguistic diversity prevents a jointly trained tokenizer from serving all languages equitably (§ 3).
- **`MoTo` models.** We train 1B non-embedding parameter decoder-only language models with `MoTo` via language-ID-based routing, where each input is tokenized by its dedicated language tokenizer at training and inference time. `MoTo` models improve compression and parity across all twenty non-English languages (§ G) and achieve consistent multilingual performance gains while preserving English performance (O. 4), with the largest improvements on morphologically complex and non-Latin-script languages (O. 5).
- **Ablations and analysis.** We ablate per-language vocabulary size, routing strategies (oracle, regularized, and random), and tokenizer granularity (per-language vs. per-script-family). We additionally train a model combining fourteen popular preexisting tokenizers into a single superset, providing a large-scale point of comparison (O. 6, 7, and 8).

2 Background

Modern language models operate on `tokens`, discrete units of text from a vocabulary [Mielke et al., 2021]. While tokenization can operate on individual bytes or whole words, subword tokenization segments raw text into units of varying length, capturing frequent words as single tokens while decomposing rare ones into smaller pieces.

Learning algorithm. Subword tokenization algorithms learn a vocabulary from a training corpus based on frequency. Byte-pair encoding (BPE) [Gage, 1994, Sennrich et al., 2016] starts from an

initial symbol set that can tokenize any input text, such as Unicode byte codes, and iteratively merges the most frequent adjacent pair until the target vocabulary size is reached. UnigramLM [Kudo and Richardson, 2018] initializes from a large seed vocabulary and iteratively prunes the tokens whose removal increases loss on a corpus the least. Both are primarily determined by the training corpus and target vocabulary size. In this work we use BPE, though `MoTo`'s design is agnostic to this choice.

Beyond the algorithm choice, every tokenizer embodies a pipeline of interacting decisions: how to normalize Unicode codepoints, how to segment raw text into coarse units before subword learning begins, what corpus to train on, and how large the vocabulary should be. These decisions are fixed before training, making them among the most consequential and least revisited choices in the entire pipeline. Each decision encodes assumptions about the intended usage of the model, but could break down when applied across typologically diverse languages and scripts. Below, we review the four axes along which these assumptions create challenges in multilingual settings, then describe how `MoTo` addresses each as an explicit, per-language design choice.

Pre-tokenization. Before learning, most tokenizers split text into coarse segments based on predefined rules. This is applied identically during model training and inference and can substantially affect model performance [Schmidt et al., 2024]. The GPT-2 regex [Radford et al., 2019] and its GPT-4 extension [OpenAI et al., 2024] have become de facto standards. Both encode English-specific assumptions: The contraction pattern `(?i:'s|'t|'re|'ve|'m|'ll|'d)` splits apostrophe-separated sequences as separate tokens, which is linguistically motivated for English (*don't* → `don` , `'t`. However, these rules can be problematic for other languages; for example, the `'d` rule would split the grammatical suffixes "-de/-da" in Turkish: *Türkiye'de* ("in Turkey") is split as `Türkiye`, `'d`, `e`. For Chinese and Japanese, the whitespace splitting assumption fails more fundamentally since these scripts do not use whitespace delimiters; without explicit character-level boundaries, BPE merges could then span entire sequences, which is intractable during tokenizer training.

Number handling is also not straightforward: digits can be grouped into fixed-size chunks (e.g., groups of three), segmented from left to right (`202`-`6`), as in the Llama-3 tokenizer, from right to left (`2`-`026`), as in Aya, or treated individually (`2`-`0`-`2`-`6`) as in Gemma-2. These choices are reported to affect arithmetic performance [Singh and Strouse, 2024], while the appropriate segmentation also depends on whether numbers represent quantities, dates, or identifiers [Garreth Lee and Wolf, 2024]. Finally, Unicode boundary handling matters for scripts where a base character and its combining marks must be kept together as an atomic unit because splitting them would produce byte sequences that do not form a valid Unicode code point.

Unicode normalization. The same glyph can be encoded as multiple distinct codepoint sequences, introducing surface variations that are visually invisible but can change tokenization drastically. While Unicode normalization could resolve this ambiguity, this choice is not always lossless. Most popular tokenizers, (e.g., Llama-3, Gemma-2, and GPT-2) apply no normalization. Qwen-3 and Aya apply NFC, which resolves canonical codepoint variations while preserving semantic distinctions. NFKC, used by XGLM [Lin et al., 2022], conflates visually similar but semantically distinct characters, destroying orthographic distinctions that are meaningful in Arabic, Farsi, and Japanese. NFC is safer but with a nuance: combining marks such as shadda (U+0651) and kasra (U+0650) in Arabic interact under canonical ordering in ways that can reorder diacritic sequences, though this does not change meaning. We find NFC to be the best choice empirically across all 21 languages in early ablations.

Frequency-driven vocabulary allocation. Languages and constructions that appear more frequently in the tokenizer-training corpus receive dedicated vocabulary entries, while underrepresented languages are left closer to the byte level, requiring more tokens to express the same content. This is particularly consequential for agglutinative languages such as Turkish and Hungarian, or root-pattern languages like Arabic where the same root appears in many distinct surface forms across inflectional paradigms; BPE may merge across morpheme boundaries inconsistently depending on which surface forms are most frequent, producing different segmentations for semantically related words. Moreover, the most likely segmentation of a string varies with language, since token frequencies differ across languages [Meister et al., 2026b].

Although morphologically informed tokenizers have been proposed [Jabbar, 2024, Asgari et al., 2025, Bayram et al., 2026], there is no strong evidence at scale that enforcing morphological boundaries improves downstream performance. Over-segmentation increases token count and can hurt rather

than help. Ideally, tokens should carry roughly uniform information content across languages and morphological structures, rather than the arbitrary boundaries of any single linguistic theory.

Vocabulary size. Vocabulary size controls the embedding parameter budget and interacts with almost every other tokenizer design choice. A larger vocabulary reduces fertility (number of tokens produced for a given string) and allows more linguistically coherent tokens, but does not guarantee multilingual coverage if the tokenizer-training corpus lacks diversity. Moreover, scaffold tokens [Lian et al., 2024], undertrained entries or glitch tokens [Li et al., 2024, Land and Bartolo, 2024], vocabulary items that appear rarely during model training, can cause unpredictable model behavior at inference. In multilingual settings a further confound arises: a language that dominates the tokenizer-training corpus will dominate the vocabulary regardless of size, crowding out underrepresented languages regardless of how large $|\mathcal{V}|$ is set.

2.1 MoTo Tokenizer Design

Instead of requiring a single monolithic tokenizer across languages, `MoTo` treats tokenizer design choices on a per-language basis. This allows the tokenizer to better respect well-defined typographical conventions without imposing English-specific assumptions. As a starting point, we train separate BPE tokenizers for 21 languages: English (`fw_edu`) and those in FineWeb2-HQ [Messmer et al., 2025] chosen to span diverse scripts and morphological typologies. These include: Germanic (German (`deu_Latn`), Dutch (`nld_Latn`), Swedish (`swe_Latn`), Danish (`dan_Latn`)), Romance (French (`fra_Latn`), Spanish (`spa_Latn`), Italian (`ita_Latn`), Portuguese (`por_Latn`)), Slavic (Polish (`pol_Latn`), Czech (`ces_Latn`), Russian (`rus_Cyrl`)), Austronesian (Indonesian (`ind_Latn`)), and Austroasiatic (Vietnamese (`vie_Latn`)) language families, alongside Hungarian (`hun_Latn`), Turkish (`tur_Latn`), Arabic (`arb_Arab`), Farsi (`fas_Arab`), Greek (`ell_Grek`), Chinese (`cmn_Hani`), and Japanese (`jpn_Jpan`). Together, this selection covers agglutinative, fusional, isolating, and root-pattern morphological typologies across Latin, Cyrillic, Arabic, Greek, and logographic scripts.

We evaluate each setting with `TokEval` [Meister, 2026] for cross-lingual parity for every language in Appendix G. We discuss further design choices below:

Language-aware pre-tokenization. Rather than using a single global regex, we assign each language a custom pre-tokenization pattern inspired by Boundless BPE [Schmidt et al., 2025], while preserving whitespace and punctuation splits. For example, Latin-script languages use a case-aware split that preserves apostrophe-joined forms as single pre-tokens important for French (*j'ai*, *l'école*), Italian (*dell'a*), and others where elision is grammatically meaningful as demonstrated in Figure 2. One can further extend this design to use Script-BPE [Land and Arnett, 2025]. `MoTo` is agnostic to the choice of pre-tokenizer; the script-aware patterns are a default that can be replaced without modifying the rest of the pipeline.

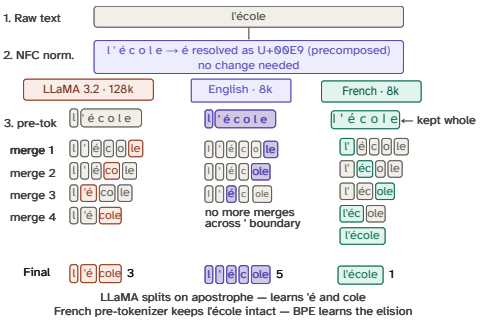

Figure 2: Tokenizing "l'école" with three BPE tokenizers with different pre-tokenization rules and vocabularies. Llama and the English tokenizer split on the apostrophe, preventing any merge across the elision boundary. The French tokenizer preserves *l'école* as a single pre-token, allowing BPE to learn the elision *l'* as a unit and compress the word to a single token. Highlighted tokens show the pair merged at each step.

Normalization. We apply NFC normalization uniformly across all languages. Although Arabic and Farsi have combining mark interactions that require care under NFC, our experiments show that applying NFC uniformly outperforms omitting normalization for these languages. The reduction in codepoint variation outweighs the rare combining mark reordering edge cases, which does not change meaning.

Initial vocabulary and vocabulary size. Beyond bytes, we seed Chinese and Japanese vocabularies with the most frequent Han, Hiragana, and

Katakana characters covering 99.9% of the respective tokenizer training corpus, ensuring single tokens for common characters rather than split byte sequences across Unicode code points. We use left-to-right three digit number tokenization, and add all integers in $[0, \ldots, 999]$. Unless otherwise stated, target vocabulary size is 8,000 (8K) per language.

Superset composition. The 21 individual tokenizers are combined into a single superset vocabulary by taking the union of all token strings based on their UTF-8 byte representations following Altıntaş et al. [2026]. The upper bound on superset size is $21 \times |\mathcal{V}|$, but languages sharing a script contribute overlapping subwords, reducing the effective cost per language. The resulting superset vocabulary sizes for each `MoTo` setting are reported in Table 1. For the 21 8K-token vocabularies we train, $|\mathcal{V}_{\mathcal{S}}|$ is below 120K. Each tokenizer maintains a mapping from its local token IDs to the global superset IDs (see Figure 1, left), we additionally reindex the tokenizers to eliminate additional lookup.

`MoTo` routing. During training, each input is routed to its dedicated tokenizer based on its language ID, tokenized locally, and then mapped to the shared superset vocabulary space before being passed to the model. At inference, the model predicts token IDs in the superset space. Our routing scheme requires no modification to the model architecture, only the embedding and unembedding matrices reflect the superset vocabulary size. When the language ID is unavailable, routing can fall back to fastText-based language identification [Joulin et al., 2016] or a base tokenizer.

`MoTo` setting. Unless otherwise stated, we refer to the combination of language-specific pre-tokenization, NFC normalization, and 8K per-language vocabulary as the default `MoTo` configuration. The full pipeline is illustrated in Figure 1 right.

3 A Controlled Study on Multilingual Addition

We first study modular tokenization on a controlled synthetic task of multilingual addition across 21 languages[1]. Given two numbers expressed as words in a target language, the model must complete the equation, e.g. "eight hundred and forty-five+five hundred and forty-five=" and produce "one thousand, three hundred and ninety". Generated in parallel across all 21 languages using the num2words package and a digit-only split, the task is minimal and fast to iterate on with small scale models, yet exposes rich linguistic diversity in number systems from script variation (Arabic, Chinese, Japanese) to numeral compounding conventions (e.g. German, Italian).

We train a shallow decoder-only transformer ($d = 256$, 4 layers, 4 attention heads, without weight-tying) with approximately 3.1M non-embedding parameters for 5000 steps (Appendix C). Our `MoTo` setting for this controlled experiment uses a lightweight superset of 1,999 tokens (256 byte tokens and at most 100 merges learned independently per language) with ID-based routing, a scaled-down instantiation of the design described in §2.1. We compare `MoTo` against three baselines: *mBPE*: a monolithic multilingual tokenizer trained jointly on all languages at 2K and 42K vocabulary sizes; *Llama*, the Llama-3.2 [Grattafiori et al., 2024] tokenizer with 128,256 tokens; and *ByT5* [Xue et al., 2022] a byte-level baseline with 256 tokens.

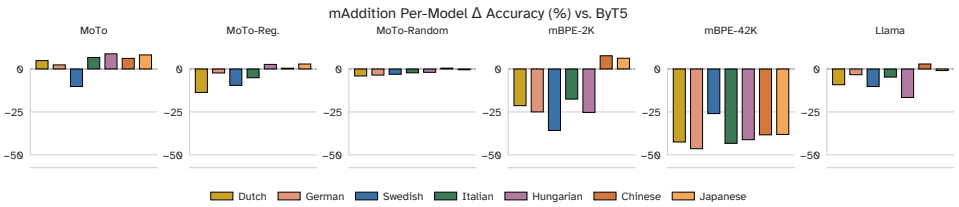

Figure 3: Per-language accuracy delta relative to ByT5 on multilingual addition, shown for the seven languages with the largest cross-tokenizer variance. `MoTo` and `MoTo`-Reg. match or exceed ByT5, while jointly trained tokenizers (mBPE) degrade substantially for languages with complex numeral or morphological structure.

O.3 *Even on a perfectly parallel dataset, linguistic diversity prevents a jointly trained tokenizer from serving all languages equitably, motivating a modular design.*

[1]Available at r-three-moto/multilingual-addition.

Table 1: Summary of all MoTo tokenizer configurations and baselines.

Configuration	Pre-tokenizer	Granularity	Vocab size
MoTo	language-specific + NFC	per language, 8K	119,716
MoTo-GPT4	GPT-4 + NFC	per language, 8K	113,764
MoTo-16K	language-specific + NFC	per language, 16K	245,153
MoTo-Script	language-specific + NFC	per script family	165,022
MoTo-Llama-16K	language-specific + NFC	per language, 16K, Llama for English	323,711
Parity-Aware BPE	—	joint but parity-aware, 128K	128,000
mBPE-GPT4	GPT-4	joint, 128K	128,000
mBPE-Boundless	Boundless BPE	joint, 128K	128,000
ByT5	—	byte-level	259
Llama	GPT-4	—	128,256
Llama-BPE-Drop.	GPT-4	—	128,256

Despite a superset of only 1,999 tokens, MoTo achieves the highest average accuracy with the largest gains on languages with complex numeral morphology (Figure 3), while still performing best on others (Figure 6). Both mBPE variants underperform on seven languages (Figure 3): languages such as Hungarian and Italian form compound number words, and under a jointly trained tokenizer these long forms consume a disproportionate share of merge operations at the expense of general linguistic structure.

How sensitive is MoTo to routing quality? Since correct language-aware routing is central to MoTo's design, we additionally evaluate two routing ablations to assess its importance: **MoTo-Reg.**, which introduces noise by routing 10% of inputs to a random tokenizer, and **MoTo-Random**, which routes all inputs randomly and serves as a lower bound on routing quality. MoTo-Reg and MoTo-Random perform comparably with ByT5, confirming that gains are driven by language-aware routing (Figures 3 and 6).

4 Multilingual Language Model Pre-training on FineWeb2 with MoTo

Having established MoTo's effectiveness in a controlled setting, we now evaluate it at a realistic scale. We train a Llama 3.2 [Grattafiori et al., 2024] style decoder-only model with approximately one billion non-embedding parameters from scratch on a 100B-token multilingual corpus of approximately 40B English tokens from FineWeb-Edu [Lozhkov et al., 2024] and 60B tokens across 20 languages from FineWeb2-HQ [Messmer et al., 2025] with sampling rates set following the UniMax strategy [Chung et al., 2023]. Full training details and data composition are reported in Appendix C.

We evaluate the downstream performance on various benchmarks suitable for base models and report performance gains relative to the Llama-3 tokenizer-model baseline, trained on the same data and compute budget. As we train a single model per setting, error bars do not reflect seed variance instead capture test-set sampling uncertainty. For each task, we compute it from the per-example differences d_i between the two models' outcomes on the same example, $\mathrm{SE} = \mathrm{std}(d)/\sqrt{n}$, using the per-sample logs of `lm-eval-harness` [Gao et al., 2023]. We combine standard errors across tasks within a language, and across languages for the average, assuming independence. Full results on a diverse set of benchmarks are presented in Appendix D.

Tokenization. The tokenizer is the only component that varies across settings. We use MoTo with language-specific pre-tokenizer and 8K per-language vocabulary as our primary configuration, constructed as described in §2.1, with inputs routed to their dedicated language tokenizer via document language ID and mapped to the shared superset vocabulary of 119,716.

Baselines. We compare MoTo against three jointly trained multilingual tokenizers: *mBPE-GPT4* and *mBPE-Boundless*, which differ only in their pre-tokenization regex (GPT-4 and Boundless BPE [Schmidt et al., 2025], respectively), and *Parity-Aware BPE* [Foroutan et al., 2025]. We also include the *Llama* 3.2 tokenizer with and without BPE-Dropout [Provilkov et al., 2020], and a byte-level baseline as in ByT5 [Xue et al., 2022]. Table 1 summarizes all configurations; further design ablations are introduced alongside their results.

4.1 Results

We report per-benchmark accuracy differences (in percentage points) relative to our Llama-3 tokenizer-model. We choose Llama's tokenizer as baseline because it is of comparable vocabulary size to `MoTo` and its multilingual coverage. All models are trained on identical data and compute.

O.4 *`MoTo` retains English performance despite allocating only 8K vocabulary entries, a fraction of Llama's vocabulary budget.*

Despite dedicating only 8K vocabulary entries to English, `MoTo` matches Llama on English commonsense reasoning benchmarks (Table 4), suggesting that a compact language-specific vocabulary can be on par with a much larger monolithic one. This may also reflect a general model improvement from exposure to larger amounts of diverse multilingual data due to compression gains during training.

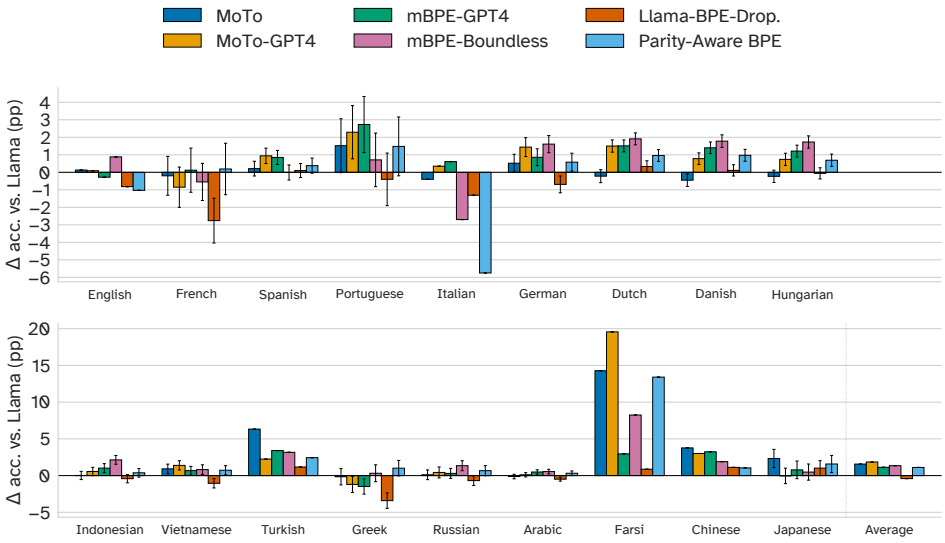

Figure 4: **Main results per language.** Accuracy difference (pp) from the Llama-3 baseline, averaged over each language's tasks; the rightmost group is the average over languages. `MoTo` and `MoTo`-GPT4 use per-language tokenizers with oracle routing. mBPE-GPT4, mBPE-Boundless, and Parity-Aware BPE are single multilingual BPE tokenizers, and Llama-BPE-Drop. is the Llama tokenizer with BPE dropout.

O.5 *Multilingual tokenization helps across the board, but non-Latin scripts and several smaller-share languages (Turkish, Vietnamese, Farsi) benefit the most.*

`MoTo` and the mBPE baselines improve over the Llama baseline across most languages (Figure 4). This suggests that two things help: training the tokenizer on the same distribution as the language model, and segmenting each language with a vocabulary built for it. We observe the largest gains with `MoTo` models on non-Latin scripts such as Farsi (Figure 4), Chinese, and Japanese (Figures 4, 12, 13) or smaller-share languages like Turkish and Vietnamese (Figures 4, 10). Improvements in high-resource Latin languages, like Italian, German, Spanish, and French remain modest.

O.6 *`MoTo`'s superset construction keeps embedding table growth sublinear in the number of languages while allowing for adaptive vocabulary expansion.*

A natural concern with modular tokenization is the cost of maintaining multiple vocabularies: naïvely, K language-specific tokenizers of size $|\mathcal{V}|$ would require an embedding table of size $K \cdot |\mathcal{V}|$. However, MoTo's superset vocabulary construction amortizes this cost and the effective vocabulary grows sublinearly with the number of languages. As seen in Table 1 combining 21 language-specific vocabularies of size 8K yields an effective superset of around 120K tokens. We observe non-trivial improvements with larger vocabulary variants (`MoTo`-16K and `MoTo`-Llama-16K) over `MoTo` with 8K vocabularies in Figure 5.

O.7 *Modular tokenization enables targeted extension of existing tokenizers without retraining the full vocabulary.*

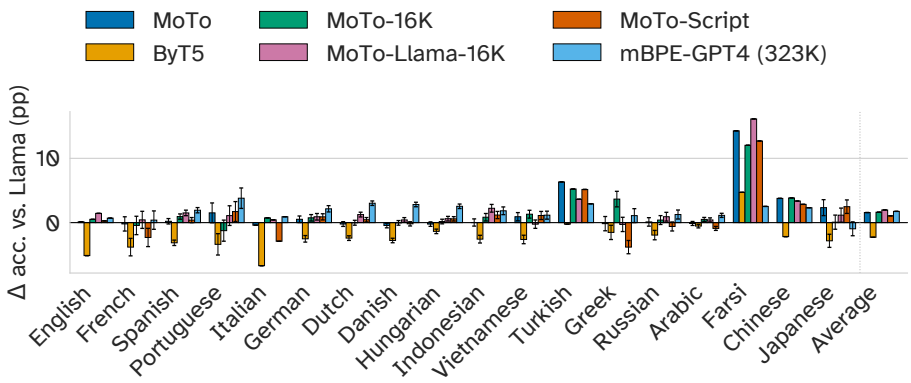

Figure 5: **Vocabulary size and granularity.** Per-language accuracy difference (pp) from the Llama-3 baseline, averaged over each language's tasks; the rightmost group is the average over languages. `MoTo` uses 21 per-language 8K BPE tokenizers; `MoTo-16K` uses 16K per-language tokenizers; `MoTo-Llama-16K` adds the Llama-3 tokenizer to the 16K per-language tokenizers as English tokenizer. ByT5 is byte-level (one token per byte), and mBPE-GPT4 (323K) is a single 323K BPE tokenizer trained jointly on all 21 languages. All `MoTo` variants use oracle routing.

Beyond raw performance, our modular design offers a practical advantage over monolithic multilingual tokenizers. Training a single large multilingual vocabulary is notoriously difficult as shown in our multilingual addition example (O. 3). For example, while training early mBPE tokenizers, we observed that some languages were overrepresented in the vocabulary. Balancing token allocations across languages, avoiding fertility disparities, and auditing a 100K+ vocabulary for artifacts or script bleed requires significant engineering effort [Rust et al., 2021]. `MoTo` mitigates this problem, as each tokenizer could be trained independently, making quality control substantially more tractable.

O.8 *`MoTo` could benefit from alternative routing strategies.*

Superset-Albert. A natural question is whether the benefits of modular composition extend to preexisting tokenizers. We construct **Superset-Albert**, a superset of fourteen off-the-shelf widely used tokenizers studied in TokSuite [Altıntaş et al., 2026] with a total vocabulary of 851,586 tokens with ALBERT-style factorized embeddings [Lan et al., 2020]. As there is no natural language-ID signal, we train this `MoTo` variant with random routing. We discuss the potential of routing and additional baselines for this setting in § E.

5 Related Work

Modular NLP & multilinguality. Our work builds on the rich literature on modular architectures for multilingual generalization. Mixture-of-Expert architectures route inputs to specialized sub-networks [Shazeer et al., 2017]. This has been leveraged for multilingual adaptation through language-specific adapters that specialize pretrained models for individual languages without full fine-tuning [Pfeiffer et al., 2020, Ansell et al., 2022, Üstün et al., 2022]. Our framework (`MoTo`) extends this principle to the tokenizer level.

Beyond subword tokenization. While byte-level transformers [Xue et al., 2022, Minixhofer et al., 2025] and pooling-based latent tokenization [Yu et al., 2023, Pagnoni et al., 2025, Hwang et al., 2025] offer alternatives; they do not necessarily guarantee multilingual fairness due to the byte-premium problem [Arnett et al., 2024] that disproportionately burdens non-Latin-script languages. Feher et al. [2025] propose dynamic tokenization with a hyper-network. Vocabulary-free approaches [Mofijul Islam et al., 2022], and end-to-end tokenization via reinforcement learning [Dauncey and Wattenhofer, 2026] explore adjacent directions but lack evidence at scale.

Stochastic approaches such as subword regularization [Kudo, 2018], BPE-Dropout [Provilkov et al., 2020], StochasTok [Sims et al., 2026] and its successor [Steger et al., 2026] introduce variability during training to improve robustness, and Zheng et al. [2025] show that models can handle non-canonical tokenizations at inference time.

Tokenizer design. Several works have relaxed individual constraints in tokenizer training. Boundless BPE [Schmidt et al., 2025] and SuperBPE [Liu et al., 2025] show performance gains when tokenizers

are allowed to learn merges across whitespace boundaries. Picky BPE [Chizhov et al., 2024] and BPE-Knockout [Bauwens and Delobelle, 2024] refine the vocabulary to eliminate redundant entries.

Tokenization and multilingual generalization. Tokenization has been shown to have consequences for multilingual performance [Rust et al., 2021, Arnett et al., 2025, 2024, Wegmann et al., 2025, Ali et al., 2024, Altıntaş et al., 2026]. Ahia et al. [2023] show that tokenization inequity manifests itself as higher fertility for morphologically rich and non-Latin-script languages, higher inference costs, and degraded downstream performance. Foroutan et al. [2025] introduce parity-based regularization to BPE learning. Schmidt et al. [2024] argue that compression is an insufficient proxy for tokenizer quality, and draw attention to pre-tokenization. Meister et al. [2026a] show that tokenizer choice matters especially for low-resource languages, and that the tokenizer properties associated with downstream performance differ across languages. Abagyan et al. [2025] show that a single multilingual tokenizer trained on many languages can achieve language plasticity in the downstream tasks.

Linguistic preprocessing tools. A rich tradition of language-specific NLP tools encodes deep linguistic knowledge for morphological analysis, normalization, and segmentation: Zemberek [zem, 2024] for Turkish, CamelTools [Obeid et al., 2020] and Farasa [Abdelali et al., 2016] for Arabic, Hazm [Research] and Parsivar [Mohtaj et al., 2018] for Farsi, Morfeusz [Woliński, 2014] for Polish, MorphoDiTa [Straková et al., 2014] for Czech, pymorphy3 [no-plagiarism, 2025] for Russian, HuSpaCy [Orosz et al., 2022, 2023] for Hungarian, and spaCy [Honnibal et al., 2020] for a range of languages. Morphologically informed tokenizers have attempted to incorporate this knowledge directly into subword learning [Jabbar, 2024, Asgari et al., 2025, Bayram et al., 2026, Zhu et al., 2025], though evidence for downstream gains remains mixed and limited to certain languages.

6 Limitations & Future Work

We train one model per setting, so uncertainty reflects test-set sampling rather than seed variance.

Routing. Our current routing operates at the sample level, assigning a single tokenizer to each sample based on its language ID. This coarse-grained approach may be suboptimal for code-switched or mixed-script inputs. While document-level language identification is well established [Joulin et al., 2016], several questions remain open: how to select a tokenizer for an unknown language, whether routing should operate at finer granularity, and whether routing decisions can be learned end-to-end during training. We discuss these challenges further in § E.

Subword sharing. Currently, `MoTo` allows modular design and segmentation, meaning that embeddings for subwords shared across languages are learned from signals from both. While sharing could benefit low-resource languages by allowing them to leverage representations learned from higher-resource languages, it could also introduce interference when shared subwords have different linguistic roles or distributions across languages. Understanding when cross-lingual sharing helps or hurts, and how it should be controlled, remains an open direction for future work.

Different allocations. Currently, `MoTo` allocates an equal share of the total vocabulary to each tokenizer. However, this allocation is a design choice rather than a constraint: vocabulary capacity could instead be distributed according to the language distribution in the model's training data, or based on other criteria such as language resource availability. Exploring alternative allocation strategies is an important direction for understanding the trade-offs between balanced multilingual coverage and overall model performance.

Instruction following. We currently evaluate `MoTo` on 1B base models. Extending `MoTo` to instruction-tuned models and evaluating open-ended generation quality across languages remains an interesting and largely unexplored direction.

7 Conclusion

Multilingual tokenization is a largely overlooked component in language model development, where English-dominant design choices propagate into suboptimal downstream performance on multilingual tasks. In this paper we show that these shortcomings are inevitable even with multilingual tokenizers trained jointly on the target corpus. Instead, we propose `MoTo` and show that treating the tokenizer as a modular, composable artifact yields consistent multilingual gains while preserving English performance, with no modification to the underlying model architecture.

Acknowledgments

We are grateful to Chris J. Maddison and Marco Ciccone for fruitful discussions and constructive feedback throughout this project. We would like to thank the members of the r-three lab for feedback at several stages of development. We thank Sven Elflein and Noah Juravsky for feedback on this manuscript. We thank Ahmadreza Jeddi and Malikeh Ehghaghi for clarifications on Farsi and Arabic, Boglárka Ecsedi on Hungarian, Haokun Liu on Chinese, Marco Ciccone on Italian, and Utku Ege Tuluk on Japanese. Gül Sena would like to thank İdil Defne Çekin, Hazal Çintaş, Azmi Ozan Ateş, and Özgem Elif Acar for their continued support.

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, the Digital Research Alliance of Canada, and companies sponsoring the Vector Institute. The authors do not have any financial conflict of interest.

References

Zemberek-nlp, 2024. URL https://github.com/ahmetaa/zemberek-nlp.

Diana Abagyan, Alejandro R. Salamanca, Andres Felipe Cruz-Salinas, Kris Cao, Hangyu Lin, Acyr Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. One tokenizer to rule them all: Emergent language plasticity via multilingual tokenizers, 2025. URL https://arxiv.org/abs/2506.10766.

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. Farasa: A fast and furious segmenter for Arabic. In John DeNero, Mark Finlayson, and Sravana Reddy, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-3003. URL https://aclanthology.org/N16-3003/.

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. Do all languages cost the same? tokenization in the era of commercial language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9904–9923, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.614. URL https://aclanthology.org/2023.emnlp-main.614/.

Mistral AI. mistral-common. https://github.com/mistralai/mistral-common, 2024.

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

Sawsan Alqahtani, Mir Tafseer Nayeem, Md Tahmid Rahman Laskar, Tasnim Mohiuddin, and M Saiful Bari. Stop taking tokenizers for granted: They are core design decisions in large language models. In Vera Demberg, Kentaro Inui, and Lluís Marquez, editors, *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8432, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.394. URL https://aclanthology.org/2026.eacl-long.394/.

Gül Sena Altıntaş, Malikeh Ehghaghi, Brian Lester, Fengyuan Liu, Wanru Zhao, Marco Ciccone, and Colin Raffel. TokSuite: Measuring the impact of tokenizer choice on language model behavior. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=vIZz7LvObC.

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.125. URL https://aclanthology.org/2022.acl-long.125/.

Catherine Arnett, Tyler A. Chang, and Benjamin Bergen. A bit of a problem: Measurement disparities in dataset sizes across languages. In Maite Melero, Sakriani Sakti, and Claudia Soria, editors, *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages @ LREC-COLING 2024*, pages 1–9, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.sigul-1.1/.

Catherine Arnett, Tyler A. Chang, Stella Biderman, and Ben Bergen. Explaining and mitigating crosslingual tokenizer inequities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0Y7AxxNCYh.

Ehsaneddin Asgari, Yassine El Kheir, and Mohammad Ali Sadraei Javaheri. Morphbpe: A morpho-aware tokenizer bridging linguistic complexity for efficient llm training across morphologies. *CoRR*, abs/2502.00894, February 2025. URL https://doi.org/10.48550/arXiv.2502.00894.

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

Thomas Bauwens and Pieter Delobelle. BPE-knockout: Pruning pre-existing BPE tokenisers with backwards-compatible morphological semi-supervision. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5810–5832, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.324. URL https://aclanthology.org/2024.naacl-long.324/.

M. Ali Bayram, Ali Arda Fincan, Ahmet Semih Gümüş, Sercan Karakaş, Banu Diri, Savaş Yıldırım, and Demircan Çelik. Tokens with meaning: A hybrid tokenization approach for turkish, 2026. URL https://arxiv.org/abs/2508.14292.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Pavel Chizhov, Catherine Arnett, Elizaveta Korotkova, and Ivan P. Yamshchikov. BPE gets picky: Efficient vocabulary refinement during tokenizer training. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16587–16604, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.925. URL https://aclanthology.org/2024.emnlp-main.925/.

Hyung Won Chung, Xavier Garcia, Adam Roberts, Yi Tay, Orhan Firat, Sharan Narang, and Noah Constant. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=kXwdL1cWOAi.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269. URL https://aclanthology.org/D18-1269/.

Sam Dauncey and Roger Wattenhofer. You can learn tokenization end-to-end with reinforcement learning, 2026. URL https://arxiv.org/abs/2602.13940.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Konstantin Dobler and Gerard de Melo. FOCUS: Effective embedding initialization for monolingual specialization of multilingual models. In Houda Bouamor, Juan Pino, and Kalika Bali,

editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13440–13454, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.829. URL https://aclanthology.org/2023.emnlp-main.829/.

Darius Feher, Ivan Vulić, and Benjamin Minixhofer. Retrofitting large language models with dynamic tokenization. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29866–29883, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1444. URL https://aclanthology.org/2025.acl-long.1444/.

Negar Foroutan, Clara Meister, Debjit Paul, Joel Niklaus, Sina Ahmadi, Antoine Bosselut, and Rico Sennrich. Parity-aware byte-pair encoding: Improving cross-lingual fairness in tokenization, 2025. URL https://arxiv.org/abs/2508.04796.

Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994. URL http://www.pennelynn.com/Documents/CUJ/HTML/94HTML/19940045.HTM.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Leandro von Werra Garreth Lee, Guilherme Penedo and Thomas Wolf. From digits to decisions: How tokenization impacts arithmetic in llms, 2024. URL https://huggingface.co/spaces/huggingface/number-tokenization-blog.

Leonidas Gee, Andrea Zugarini, Leonardo Rigutini, and Paolo Torroni. Fast vocabulary transfer for language model compression. In Yunyao Li and Angeliki Lazaridou, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 409–416, Abu Dhabi, UAE, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-industry.41. URL https://aclanthology.org/2022.emnlp-industry.41/.

Charles Goddard and Fernando Fernandes Neto. Training-free tokenizer transplantation via orthogonal matching pursuit, 2025. URL https://arxiv.org/abs/2506.06607.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde,

Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro

Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020. doi: 10.5281/zenodo.1212303.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling, 2025. URL https://arxiv.org/abs/2507.07955.

Haris Jabbar. Morphpiece : A linguistic tokenizer for large language models, 2024. URL https://arxiv.org/abs/2307.07262.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2068/.

Jaap Jumelet, Leonie Weissweiler, Joakim Nivre, and Arianna Bisazza. MultiBLiMP 1.0: A massively multilingual benchmark of linguistic minimal pairs. *Transactions of the Association for Computational Linguistics*, 14:193–216, 2026. doi: 10.1162/tacl.a.600. URL https://aclanthology.org/2026.tacl-1.10/.

Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi, Luca Soldaini, Enrico Shippole, A. Feder Cooper, Aviya Skowron, Shayne Longpre, Lintang Sutawika, Alon Albalak, Zhenlin Xu, Guilherme Penedo, Loubna Ben allal, Elie Bakouch, John David Pressman, Honglu Fan, Dashiell Stander, Guangyu Song, Aaron Gokaslan, John Kirchenbauer, Tom Goldstein, Brian R. Bartoldson, Bhavya Kailkhura, and Tyler Murray. The common pile v0.1: An 8TB dataset of public domain and openly licensed text. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=DIELgiqdvJ.

Amir Hossein Kargaran, Ayyoob Imani, François Yvon, and Hinrich Schuetze. GlotLID: Language identification for low-resource languages. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6155–6218, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.410. URL https://aclanthology.org/2023.findings-emnlp.410/.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*,

pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL https://aclanthology.org/P18-1007/.

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

Viet Lai, Chien Nguyen, Nghia Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan Rossi, and Thien Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In Yansong Feng and Els Lefever, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 318–327, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-demo.28. URL https://aclanthology.org/2023.emnlp-demo.28/.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020. URL https://arxiv.org/abs/1909.11942.

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

Sander Land and Max Bartolo. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11631–11646, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.649. URL https://aclanthology.org/2024.emnlp-main.649/.

Yuxi Li, Yi Liu, Gelei Deng, Ying Zhang, Wenjia Song, Ling Shi, Kailong Wang, Yuekang Li, Yang Liu, and Haoyu Wang. Glitch tokens in large language models: Categorization taxonomy and effective detection. *Proc. ACM Softw. Eng.*, 1(FSE), July 2024. doi: 10.1145/3660799. URL https://doi.org/10.1145/3660799.

Haoran Lian, Yizhe Xiong, Jianwei Niu, Shasha Mo, Zhenpeng Su, Zijia Lin, Hui Chen, Peng Liu, Jungong Han, and Guiguang Ding. Scaffold-bpe: Enhancing byte pair encoding for large language models with simple and effective scaffold token removal, 2024. URL https://arxiv.org/abs/2404.17808.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. Few-shot learning with multilingual generative language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9019–9052, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.616. URL https://aclanthology.org/2022.emnlp-main.616/.

Alisa Liu, Jonathan Hayase, Valentin Hofmann, Sewoong Oh, Noah A. Smith, and Yejin Choi. SuperBPE: Space travel for language models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=lcDRvffeNP.

Yihong Liu, Peiqin Lin, Mingyang Wang, and Hinrich Schuetze. OFA: A framework of initializing unseen subword embeddings for efficient large-scale multilingual continued pretraining. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1067–1097, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.68. URL https://aclanthology.org/2024.findings-naacl.68/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

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

Clara Meister, Gül Sena Altıntaş, and Antoine Bosselut. Language-specific effects of tokenizer choice in multilingual language models, 2026a. URL https://arxiv.org/abs/2610.12144.

Clara Meister, Ahmetcan Yavuz, Pietro Lesci, and Tiago Pimentel. What language is this? ask your tokenizer, 2026b. URL https://arxiv.org/abs/2602.17655.

Bettina Messmer, Vinko Sabolčec, and Martin Jaggi. Enhancing multilingual LLM pretraining with model-based data selection. In Jonathan Gerber, Mark Cieliebak, Don Tuggener, and Manuela Hürlimann, editors, *Proceedings of the 10th edition of the Swiss Text Analytics Conference*, pages 31–56, Winterthur, Switzerland, May 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.swisstext-1.4/.

Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *CoRR*, abs/2112.10508, 2021. URL https://arxiv.org/abs/2112.10508.

Benjamin Minixhofer, Fabian Paischer, and Navid Rekabsaz. WECHSEL: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3992–4006, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.293. URL https://aclanthology.org/2022.naacl-main.293/.

Benjamin Minixhofer, Edoardo Ponti, and Ivan Vulić. Zero-shot tokenizer transfer. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=RwBObRsIzC.

Benjamin Minixhofer, Tyler Murray, Tomasz Limisiewicz, Anna Korhonen, Luke Zettlemoyer, Noah A. Smith, Edoardo M. Ponti, Luca Soldaini, and Valentin Hofmann. Bolmo: Byteifying the next generation of language models. *CoRR*, abs/2512.15586, 2025. doi: 10.48550/ARXIV.2512.15586. URL https://doi.org/10.48550/arXiv.2512.15586.

Md Mofijul Islam, Gustavo Aguilar, Pragaash Ponnusamy, Clint Solomon Mathialagan, Chengyuan Ma, and Chenlei Guo. A vocabulary-free multilingual neural tokenizer for end-to-end task learning. In Spandana Gella, He He, Bodhisattwa Prasad Majumder, Burcu Can, Eleonora Giunchiglia, Samuel Cahyawijaya, Sewon Min, Maximilian Mozes, Xiang Lorraine Li, Isabelle Augenstein, Anna Rogers, Kyunghyun Cho, Edward Grefenstette, Laura Rimell, and Chris Dyer, editors, *Proceedings of the 7th Workshop on Representation Learning for NLP*, pages 91–99, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.repl4nlp-1.10. URL https://aclanthology.org/2022.repl4nlp-1.10/.

Salar Mohtaj, Behnam Roshanfekr, Atefeh Zafarian, and Habibollah Asghari. Parsivar: A language processing toolkit for Persian. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1179/.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. Crosslingual generalization through multitask finetuning, 2022.

no-plagiarism. pymorphy3: Morphological analyzer for russian and ukrainian. https://github.com/no-plagiarism/pymorphy3, 2025. Continuation of pymorphy2.

num2words. num2words library - convert numbers to words in multiple languages.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. CAMeL tools: An open source python toolkit for Arabic natural language processing. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.868/.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang,

Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

György Orosz, Zsolt Szántó, Péter Berkecz, Gergő Szabó, and Richárd Farkas. HuSpaCy: an industrial-strength Hungarian natural language processing toolkit. In *XVIII. Magyar Számítógépes Nyelvészeti Konferencia*, pages 59–73, 2022.

György Orosz, Gergő Szabó, Péter Berkecz, Zsolt Szántó, and Richárd Farkas. "Advancing Hungarian Text Processing with HuSpaCy: Efficient and Accurate NLP Pipelines". In Kamil "Ekštein, František Pártl, and Miloslav" Konopík, editors, *"Text, Speech, and Dialogue"*, pages 58–69, "Cham", 2023. "Springer Nature Switzerland". ISBN 978-3-031-40498-6.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. Byte latent transformer: Patches scale better than tokens. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.453. URL https://aclanthology.org/2025.acl-long.453/.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language Model Tokenizers Introduce Unfairness Between Languages. November 2023. URL https://openreview.net/forum?id=78yDLKi95p.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.617. URL https://aclanthology.org/2020.emnlp-main.617/.

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

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. BPE-dropout: Simple and effective subword regularization. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.170. URL https://aclanthology.org/2020.acl-main.170/.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.

Roshan Research. Hazm: Persian text processing library. https://github.com/roshan-research/hazm.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*, 2011. URL https://people.ict.usc.edu/~gordon/publications/AAAI-SPRING11A.PDF.

Angelika Romanou, Negar Foroutan, Anna Sotnikova, Zeming Chen, Sree Harsha Nelaturu, Shivalika Singh, Rishabh Maheshwary, Micol Altomare, Mohamed A Haggag, Alfonso Amayuelas, et al. Include: Evaluating multilingual language understanding with regional knowledge. *arXiv preprint arXiv:2411.19799*, 2024.

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

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

Craig W Schmidt, Varshini Reddy, Chris Tanner, and Yuval Pinter. Boundless byte pair encoding: Breaking the pre-tokenization barrier. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=oPAjXGV8qQ.

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

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Anya Sims, Thom Foster, Klara Kaleb, Tuan-Duy H. Nguyen, Joseph Lee, Jakob N. Foerster, Yee Whye Teh, and Cong Lu. Stochastok: Improving fine-grained subword understanding in llms, 2026. URL https://arxiv.org/abs/2506.01687.

Aaditya K. Singh and D. J. Strouse. Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs, February 2024. URL http://arxiv.org/abs/2402.14903. arXiv:2402.14903 [cs.CL].

Sophie Steger, Rui Li, Sofiane Ennadir, Anya Sims, Arno Solin, Franz Pernkopf, and Martin Trapp. Stochasticity in tokenisation improves robustness, 2026. URL https://arxiv.org/abs/2604.16037.

Jana Straková, Milan Straka, and Jan Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-5003.pdf.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No language left behind: Scaling human-centered machine translation, 2022. URL https://arxiv.org/abs/2207.04672.

Alexey Tikhonov and Max Ryabinin. It's all in the heads: Using attention heads as a baseline for cross-lingual transfer in commonsense reasoning, 2021.

Ke Tran. From english to foreign languages: Transferring pre-trained language models, 2020. URL https://arxiv.org/abs/2002.07306.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. UDapter: Typology-based language adapters for multilingual dependency parsing and sequence labeling. *Computational Linguistics*, 48(3):555–592, September 2022. doi: 10.1162/coli_a_00443. URL https://aclanthology.org/2022.cl-3.3/.

Ahmet Üstün, Viraat Aryabumi, Zheng Yong, Wei-Yin Ko, Daniel D'souza, Gbemileke Onilude, Neel Bhandari, Shivalika Singh, Hui-Lee Ooi, Amr Kayid, Freddie Vargus, Phil Blunsom, Shayne Longpre, Niklas Muennighoff, Marzieh Fadaee, Julia Kreutzer, and Sara Hooker. Aya model: An instruction finetuned open-access multilingual language model. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15894–15939, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.845. URL https://aclanthology.org/2024.acl-long.845/.

Mathurin Videau, Badr Youbi Idrissi, Daniel Haziza, Luca Wehrstedt, Jade Copet, Olivier Teytaud, and David Lopez-Paz. Meta Lingua: A minimal PyTorch LLM training library, 2024. URL https://github.com/facebookresearch/lingua.

Anna Wegmann, Dong Nguyen, and David Jurgens. Tokenization is sensitive to language variation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10958–10983, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.572. URL https://aclanthology.org/2025.findings-acl.572/.

Marcin Woliński. Morfeusz reloaded. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1106–1111, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). URL https://aclanthology.org/L14-1597/.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022. doi: 10.1162/tacl_a_00461. URL https://aclanthology.org/2022.tacl-1.17/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yilun Yang and Yekun Chai. CodeMixBench: Evaluating code-mixing capabilities of LLMs across 18 languages. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2139–2169, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.109. URL https://aclanthology.org/2025.emnlp-main.109/.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *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, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1382. URL https://aclanthology.org/D19-1382/.

Lili Yu, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. MEGABYTE: Predicting million-byte sequences with multiscale transformers. In *Thirty-seventh*

Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=JTmO2V9Xpz.

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

Brian Siyuan Zheng, Alisa Liu, Orevaoghene Ahia, Jonathan Hayase, Yejin Choi, and Noah A. Smith. Broken tokens? your language model can secretly handle non-canonical tokenizations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=WrYWolqKh3.

Qingyang Zhu, Xiang Hu, Pengyu Ji, Wei Wu, and Kewei Tu. Unsupervised morphological tree tokenizer. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22299–22312, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1146. URL https://aclanthology.org/2025.findings-acl.1146/.

Appendix Contents

A Ethics Statement 25

B Declaration of LLM Usage 25

C Multilingual Language Model Pre-training with MoTo 25

D Full Results 25
- D.1 Multilingual Addition 25
- D.2 Multilingual Pre-training 26

E Routing 30

F Merging Per-Language Tokenizers into a Monolithic BPE Tokenizer 35

G Tokenizer Quality & Compression 38

A Ethics Statement

This work addresses tokenization inequities in multilingual language models, a problem with direct consequences for speakers of underrepresented languages, who may experience higher costs and degraded model performance due to design choices made primarily for high-resource languages. We acknowledge that our study currently covers 21 languages among many of the world's languages.

B Declaration of LLM Usage

Large Language Models (LLMs) were used for grammar checking and feedback on clarity during the writing of this paper. Throughout the development of the project LLMs were also used for assisting in ideation, exploration, coding, diagram, and figure creation.

C Multilingual Language Model Pre-training with MoTo

Model architecture. All experiments use Llama 3.2 [Grattafiori et al., 2024] architecture with approximately one billion non-embedding parameters as the base transformer architecture ($d = 2048$, 25 layers, 16 attention heads, no weight-tying).

Model training. All models are trained from scratch using the Meta Lingua framework [Videau et al., 2024] with full parameter updates in `bfloat16` precision for 100,000 steps with AdamW [Loshchilov and Hutter, 2019] optimizer. We use a cosine learning rate schedule with a peak learning rate of 10^{-3}, 2,000 warmup steps, gradient clipping at 1.0, and weight decay of 0.1. The effective batch size is 256 with a sequence length of 4,096 tokens.

Each model is trained on a single 8×H100 node. Models with effective (superset) vocabulary size around 128K (e.g., Llama, MoTo, MoTo-GPT4, mBPE) take approximately 4 days to train, while larger ablations (e.g., MoTo-16K or MoTo-Script) take approximately 5.5 days.

Training data. We train all models on a multilingual corpus combining English and 20 additional languages. English data is sampled from the 100 billion subset of FineWeb-Edu [Lozhkov et al., 2024], a high-quality educational data filtered from FineWeb [Penedo et al., 2024]; while non-English languages are sampled from FineWeb2-HQ [Messmer et al., 2025]. Sampling weights are set proportional to corpus availability following the UniMax strategy [Chung et al., 2023], with English receiving 40% of the training budget; full per-language weights are reported in Table 2.

D Full Results

D.1 Multilingual Addition

We observe consistent gains on the remaining twelve languages in Figure 6 with MoTo. In this setting, the disparity between MoTo and mBPE is smaller as these languages have simpler numeral morphology than those in Figure 3.

Table 2: Data mixture rates for pre-training following UniMax [Chung et al., 2023] sampling.

FLORES code	Language	Script	Mixture Rate	Approx. # Tokens (B/1e9)
arb_Arab	Arabic	Arabic	0.02343237066	2.46
cmn_Hani	Chinese	Hani	0.04540675214	4.76
ces_Latn	Czech	Latin	0.02480244553	2.60
dan_Latn	Danish	Latin	0.02165828697	2.27
nld_Latn	Dutch	Latin	0.02777963266	2.91
eng_Latn	English	Latin	0.4	41.94
fra_Latn	French	Latin	0.03818355607	4.00
deu_Latn	German	Latin	0.04199253405	4.40
ell_Grek	Greek	Greek	0.02336708869	2.45
hun_Latn	Hungarian	Latin	0.02471945736	2.59
ind_Latn	Indonesian	Latin	0.02693220546	2.82
ita_Latn	Italian	Latin	0.0324056371	3.40
jpn_Jpan	Japanese	Japanese	0.03553104151	3.73
fas_Arab	Farsi	Arabic	0.02056346242	2.16
pol_Latn	Polish	Latin	0.02941201046	3.08
por_Latn	Portuguese	Latin	0.03014131683	3.16
rus_Cyrl	Russian	Cyrillic	0.05001981574	5.24
spa_Latn	Spanish	Latin	0.03872227931	4.06
swe_Latn	Swedish	Latin	0.02163597654	2.27
tur_Latn	Turkish	Latin	0.02354557948	2.47
vie_Latn	Vietnamese	Latin	0.01974855103	2.07
			Total	104.86

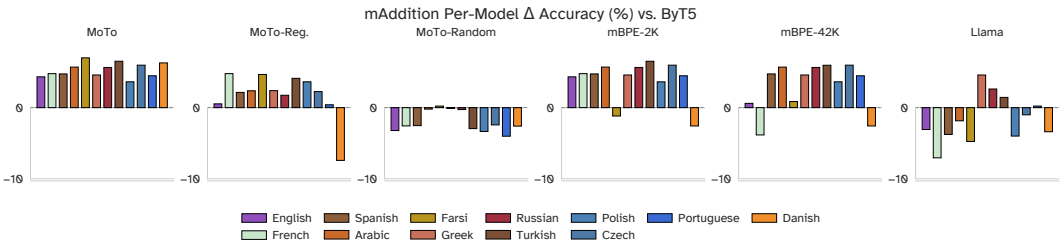

Figure 6: Per-language percentage accuracy delta relative to ByT5 tokenizer on multilingual addition for languages with small performance differences.

D.2 Multilingual Pre-training

We evaluate all models on a diverse set of multilingual and English downstream tasks suitable for base models. Not all languages are uniformly available across all benchmarks. Language coverage across all tasks is reported in Table 3. For language modeling ability we report bits-per-byte on FLORES+ across all 21 languages and accuracy on MultiBLIMP [Jumelet et al., 2026] across 17 languages. For **commonsense reasoning**, we use HellaSwag [Zellers et al., 2019], PIQA [Bisk et al., 2020], BoolQ [Clark et al., 2019], ARC Easy and Challenge [Clark et al., 2018], and CommonsenseQA [Talmor et al., 2019] in English, alongside the machine-translated multilingual HellaSwag [Lai et al., 2023] across 13 languages, XCOPA [Ponti et al., 2020, Roemmele et al., 2011] across 5 languages, and XStoryCloze [Lin et al., 2022] across 6 languages. For **natural language inference**, we evaluate on XNLI [Conneau et al., 2018] across 10 languages, PAWS-X [Yang et al., 2019] across 6 languages, and XWinograd [Muennighoff et al., 2022, Tikhonov and Ryabinin, 2021] across 6 languages. For **knowledge and language understanding**, we evaluate on canonical completion tasks of TokSuite [Altıntaş et al., 2026] across 5 languages. We also evaluate on INCLUDE-base-44 [Romanou et al., 2024], Belebele [Bandarkar et al., 2024], but find these benchmarks too challenging for 1B-parameter base models; yielding near random chance performance. We report per-language results across all benchmarks relative to the Llama baseline. Unless otherwise noted, positive values indicate improvement over Llama.

Table 3: Downstream task evaluation coverage per language. ✓indicates the language is evaluated on the corresponding benchmark group.
∗: While Chinese subset is reported in Lai et al. [2023], it is not available in lm-eval-harness [Gao et al., 2023].

Language	M-HellaSwag	XCOPA	XStoryCloze	PAWS-X	XNLI	TokSuite	Wino-X	Multi-BLIMP	Coverage
Arabic	✓		✓		✓			✓	4
Chinese	*	✓	✓	✓	✓	✓	✓		6
Czech								✓	1
Danish	✓							✓	2
Dutch	✓							✓	2
English			✓	✓	✓	✓	✓	✓	6
Farsi						✓		✓	2
French	✓			✓	✓		✓	✓	5
German	✓			✓	✓			✓	4
Greek					✓			✓	2
Hungarian	✓							✓	2
Indonesian	✓	✓	✓						3
Italian	✓	✓				✓		✓	4
Japanese				✓			✓		2
Polish								✓	1
Portuguese	✓						✓	✓	3
Russian	✓		✓		✓		✓	✓	5
Spanish	✓		✓	✓	✓			✓	5
Swedish	✓							✓	2
Turkish		✓			✓	✓		✓	4
Vietnamese	✓	✓			✓				3
Total	13	5	6	6	10	5	6	17	

Table 4: Relative accuracy difference (%) on English benchmarks compared to Llama-3 tokenizer model. Higher better.

Model	HellaSwag	ARC-Easy	ARC-Challenge	CommonsenseQA	PIQA	BoolQ	XWinograd	Avg
MoTo	$+0.30 \pm 0.31$	$+0.13 \pm 0.78$	$+0.09 \pm 1.19$	-3.03 ± 1.66	$+0.87 \pm 0.73$	$+4.38 \pm 1.02$	$+1.08 \pm 0.88$	$+0.54 \pm 0.38$
MoTo-GPT4	$+0.06 \pm 0.31$	-0.80 ± 0.74	$+1.37 \pm 1.15$	-2.86 ± 1.67	-0.16 ± 0.74	$+2.45 \pm 1.05$	$+0.22 \pm 0.89$	$+0.04 \pm 0.38$
mBPE-GPT4	$+1.33 \pm 0.31$	$+0.72 \pm 0.74$	$+0.17 \pm 1.19$	-3.68 ± 1.25	-0.16 ± 0.75	$+0.80 \pm 0.87$	$+1.60 \pm 0.86$	$+0.11 \pm 0.34$
mBPE-Boundless	$+1.34 \pm 0.31$	$+2.23 \pm 0.77$	$+1.88 \pm 1.18$	-3.11 ± 1.64	$+0.11 \pm 0.73$	$+1.29 \pm 0.99$	$+2.33 \pm 0.84$	$+0.86 \pm 0.38$
Llama-BPE-Drop.	-1.03 ± 0.30	$+0.55 \pm 0.75$	$+0.68 \pm 1.15$	-3.11 ± 1.60	-0.71 ± 0.71	$+1.93 \pm 1.00$	-0.60 ± 0.87	-0.33 ± 0.37
Parity-Aware BPE	$+1.39 \pm 0.31$	-0.29 ± 0.74	$+0.68 \pm 1.18$	-1.63 ± 1.61	-0.33 ± 0.76	$+2.24 \pm 0.97$	$+0.52 \pm 0.88$	$+0.37 \pm 0.38$

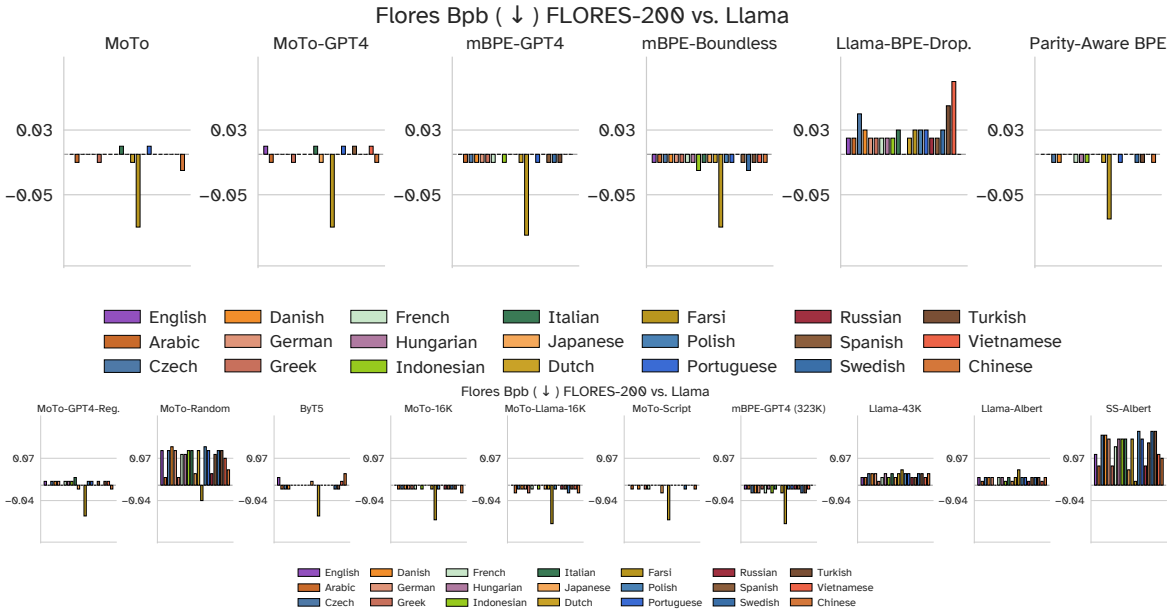

Figure 7: Per-language bits-per-byte (BPB) on FLORES+ relative to the Llama baseline (top: main configurations; bottom: ablations). Lower is better.

English performance

Language modeling *(FLORES+ bits-per-byte (BPB), Figure 7).* In terms of bits-per-bytes, `MoTo` settings and mBPE settings perform comparably to the Llama baselines with significant improvement in Farsi. We observe similar gains in Farsi in `TokSuite` (Figures 14 and 15) suggesting that Farsi is under-served by Llama tokenizer. With the distinction of BPE-Dropout baseline performing worse than Llama, all settings perform similarly.

Grammatical agreement (MultiBLiMP, Figure 8). MultiBLiMP [Jumelet et al., 2026] tests subject–verb agreement in person, number and gender with minimal pairs. A model is correct when it assigns the grammatical sentence a higher likelihood than its ungrammatical twin. We evaluate the 17 of our 21 languages that it covers. In general the absolute accuracies are saturated for MultiBLIMP.

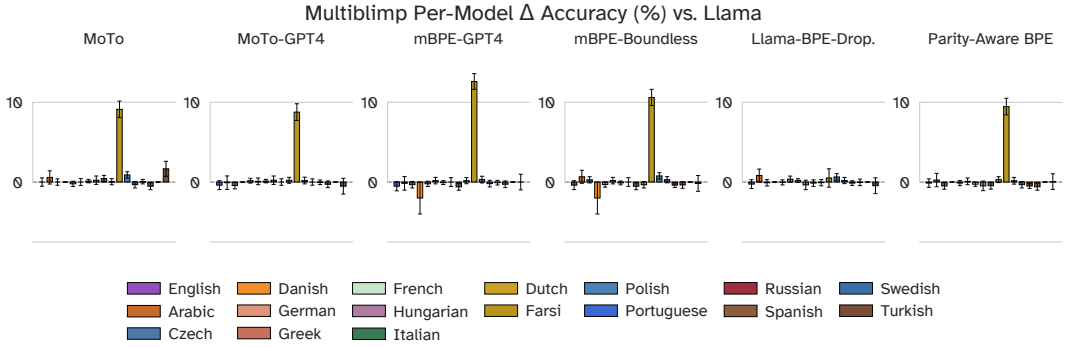

Figure 8: Per-language accuracy difference on MultiBLIMP [Jumelet et al., 2026] relative to the Llama baseline.

Multilingual commonsense reasoning (Okapi, Figure 9). `MoTo`-GPT4 shows gains across the board for all thirteen languages. Multilingual baselines perform consistently better with Boundless-BPE pre-tokenization overperforming GPT4 especially in French, Indonesian, and Portuguese.

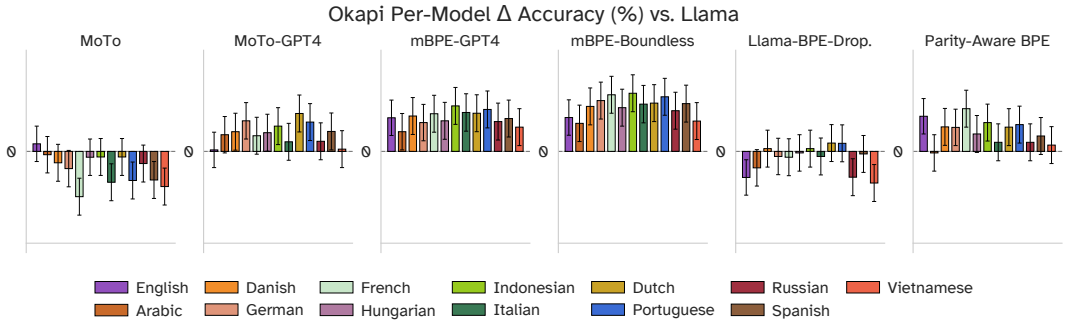

Figure 9: Per-language accuracy difference on multilingual HellaSwag (Okapi) [Lai et al., 2023] relative to the Llama baseline.

Causal commonsense reasoning (XCOPA, Figure 10). `MoTo` variants outperform the Llama baseline and mBPE across all tasks, with mBPE showing larger improvements on Indonesian.

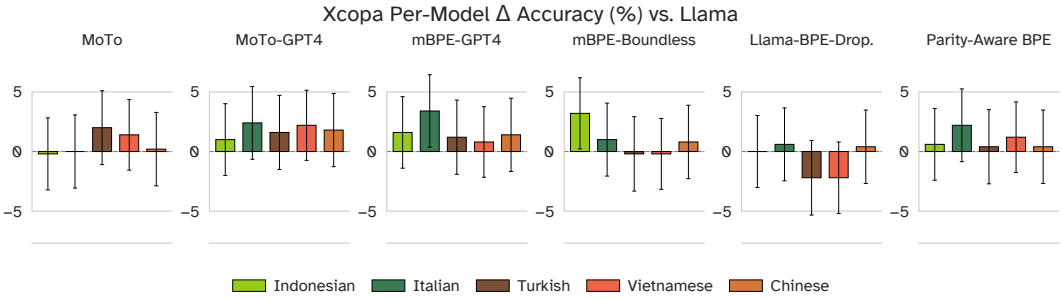

Figure 10: Per-language accuracy difference on XCOPA [Ponti et al., 2020] causal commonsense reasoning benchmark relative to the Llama baseline.

Narrative commonsense reasoning (XStoryCloze, Figure 11). MoTo variants exhibit consistent gains on Russian, Spanish, and Indonesian. mBPE-GPT4 achieves consistent gains on Spanish. Larger vocabulary variants (MoTo-16K, MoTo-Llama-16K) show additional benefits, suggesting vocabulary size matters for story understanding tasks.

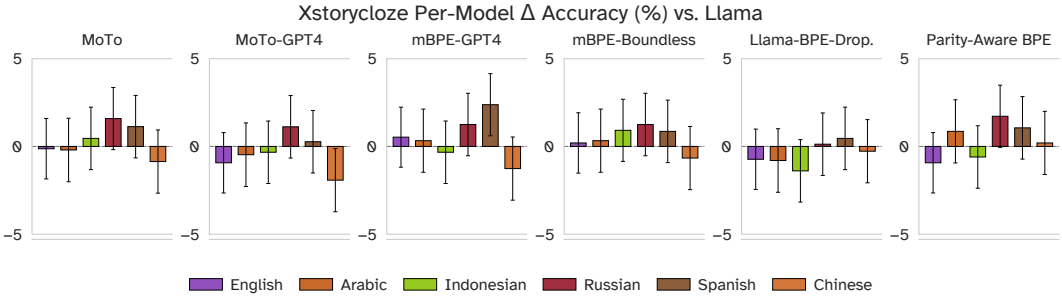

Figure 11: Per-language accuracy difference on XStoryCloze [Lin et al., 2022] relative to the Llama baseline. XStoryCloze tests narrative commonsense reasoning for story understanding.

Paraphrase identification (PAWS-X, Figure 12). MoTo models show substantial improvements in Japanese and Chinese, with modest gains on Spanish.

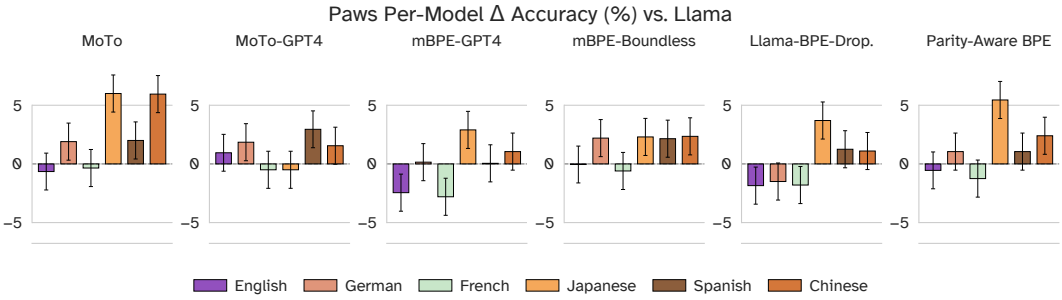

Figure 12: Per-language accuracy difference on PAWS-X [Yang et al., 2019] relative to the Llama baseline. PAWS-X tests paraphrase identification.

Coreference and pronoun resolution (XWinograd, Figure 13). MoTo variants exhibit large improvements in Portuguese, an observation we also make in HellaSwag (Figure 9). Chinese benefits from the modular design across all MoTo variants, while improvements in Japanese are modest.

Knowledge probing and NLI (TokSuite and XNLI, Figures 14 and 15). On TokSuite canonical completion tasks, Farsi shows the most consistent improvement across all MoTo variants, reflecting Llama's systematic underservice of Arabic-script languages. Chinese and Turkish also show consistent gains, with small losses on Italian. On XNLI, Greek, Turkish, and Russian improve with

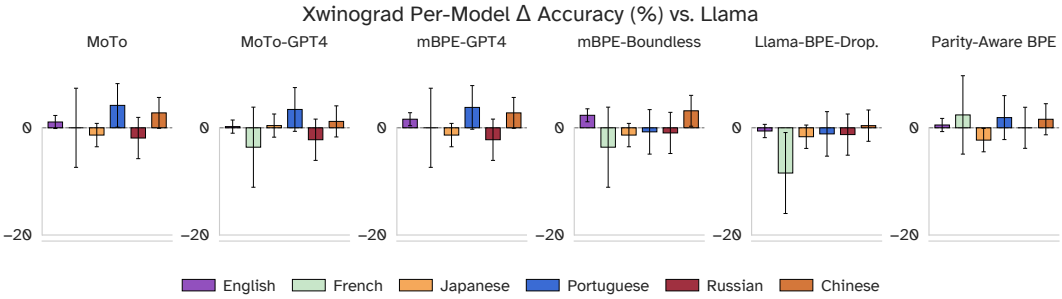

Figure 13: Per-language accuracy difference on XWinograd [Muennighoff et al., 2022, Tikhonov and Ryabinin, 2021] relative to the Llama baseline. XWinograd evaluates pronoun resolution and coreference across multiple languages.

MoTo-16K and MoTo-Reg.; however, Greek is hurt under script-level grouping (MoTo-Script) even though both mixtures utilize the same underlying Greek tokenizer.

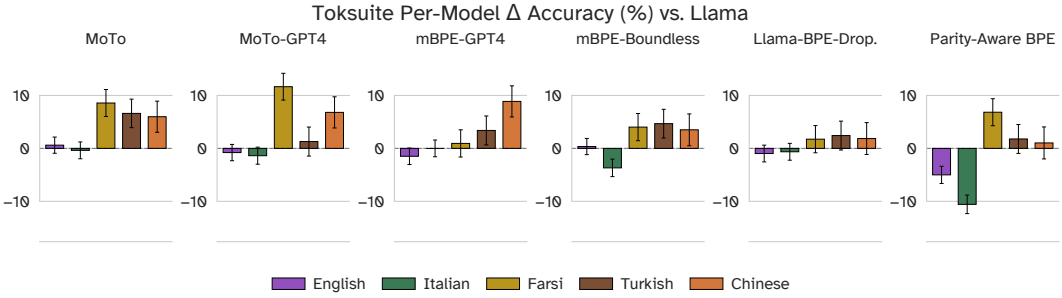

Figure 14: Ablation results on TokSuite completion tasks relative to the Llama baseline.

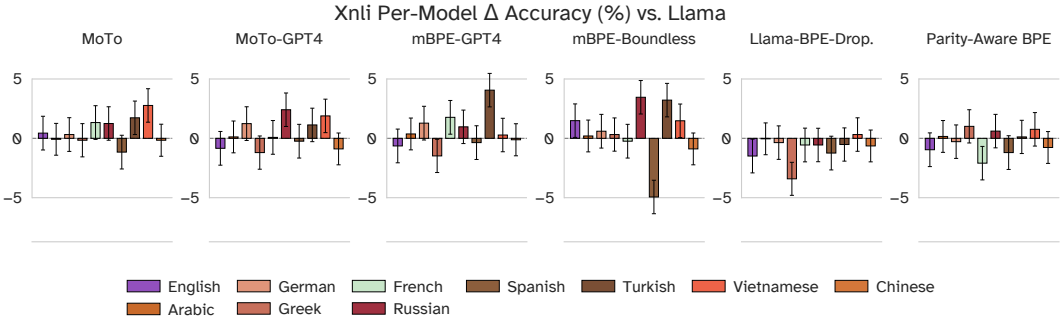

Figure 15: Ablation results on XNLI natural language inference (bottom), relative to the Llama baseline.

E Routing

In MoTo each input (a training document or an evaluation prompt) is routed to the tokenizer of its language, which we call LID routing. During pre-training, we take the language from the per-document labels that most multilingual corpora already provide, so routing adds no compute overhead to the training loop. At evaluation time, we use the benchmark's language when it is known, and otherwise predict it with GlotLID [Kargaran et al., 2023], restricted to the 21 languages in the mixture. While this is well-defined in the pre-training regime we study, where each tokenizer is trained on a single language and documents come with language labels and each benchmark has language subsets, it raises three questions: what is the additional cost of routing when labels are unavailable, how it handles mixed-language inputs, and what happens when no tokenizer is a clear match for the input.

Mixing existing tokenizers. MoTo trains its own per-language tokenizers. A natural alternative is to reuse tokenizers that already exist. Superset-Albert (SS-Albert) maps 14 off-the-shelf tokenizers from TokSuite [Altıntaş et al., 2026] into one 851K-entry vocabulary and encodes every training document with one of them, drawn at random: Aya Expanse, BLOOM, Comma, Gemma 2, GPT-2, Llama 3.2, mBERT, Phi-3, Qwen3, XGLM, ByT5, GPT-4o, Tekken and TokenMonster. A full-width table for this vocabulary would dominate the parameter count, so SS-Albert factorizes its input and output embeddings ALBERT-style ($851\text{K} \times 512 \rightarrow 2{,}048$). We compare it with two Llama 3.2 controls (Figure 16): Llama-Albert, with the same factorized embeddings and the Llama tokenizer alone, which isolates the effect of the tokenizer mixture from that of the factorization; and Llama-43K, the Llama baseline stopped after 43K of its 100K steps, which is the estimated number of steps embeddings for the Llama tokenizer are trained as part of the SS-Albert setting. We observe that all settings result in degradation and discuss implications for SS-Albert further below.

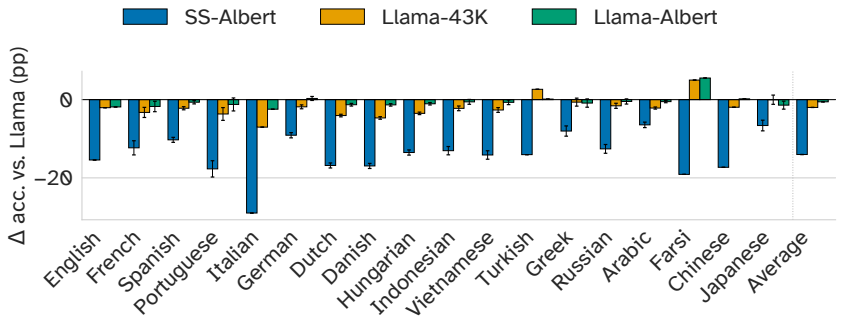

Figure 16: **Mixing existing tokenizers.** Per-language accuracy difference (pp) from the Llama 3.2 baseline, averaged over each language's tasks; the rightmost group is the average over languages. SS-Albert maps 14 existing tokenizers into one vocabulary with factorized embeddings. Llama-Albert uses the same factorized embeddings with the Llama tokenizer only. Llama-43K is the Llama baseline after 43K of its 100K training steps.

Language-ID (LID) routing has minimal overhead. For unlabeled corpora, language identification would have to run alongside training, but its cost is negligible compared to that of training. For example, GlotLID is a fastText classifier [Joulin et al., 2016] is a linear model over hashed character and word n-gram embeddings that runs on CPU. fastText [Joulin et al., 2017] classifies half a million sentences among 312K classes in under a minute on a standard multicore CPU, and GlotLID applies the same architecture to over 1,600 languages [Kargaran et al., 2023]. Language identification can therefore run by data-loader workers, in parallel with GPU training and at a negligible fraction of the cost of a forward and backward pass. Restricting predictions to the languages in the mixture, as we do, further simplifies this procedure.

Regularizing LID routing Oracle routing assumes that every document's language is known. To test whether MoTo depends on this, we train with regularized routing: each document is routed to its language's tokenizer with probability 0.9 and to a uniformly random component otherwise (MoTo-GPT4-Reg.). We also train a fully random-routing model as a lower bound (MoTo-Random). We evaluate both settings with oracle routing. Figure 17 reports each model's accuracy relative to MoTo. Regularization costs little: 1.1 points on average, and it matches or slightly exceeds MoTo on 10 of 18 languages. Random routing costs 5.2 points, with the largest drops in Farsi (-12.9) and Turkish (-10.5), languages whose tokenizers are among the most specialized in the mixture. The model therefore benefits from seeing each language in its own segmentation most of the time, but not from seeing it exclusively.

Limitations of LID routing. Document-level LID assigns one tokenizer per input, so code-switched or transliterated text could suffer in performance; or short documents could be misrouted to the tokenizer of closely related languages (e.g., the Farsi–Arabic or Spanish–Italian confusions common in short inputs). Languages outside the mixture are routed randomly or to a fallback tokenizer. Routing at a finer granularity or learning a router could address these cases.

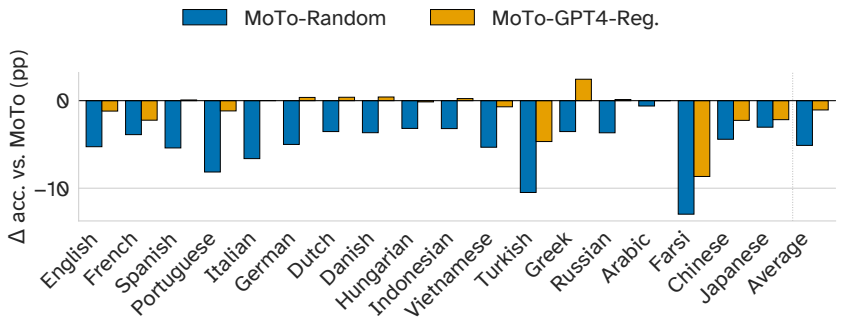

Figure 17: **Routing ablation.** Per-language accuracy difference (pp) from `MoTo`, averaged over each language's tasks; the rightmost group is the average over languages. `MoTo`-GPT4-Reg. routes each training document to its language's tokenizer with probability 0.9 and to a random component otherwise; `MoTo`-Random always routes at random. All models are evaluated with oracle routing.

Code-switching. Assigning one language per input raises the question of how to route mixed-language text. We evaluate two code-switched settings: `TokSuite` [Altıntaş et al., 2026] examples in Chinese, Farsi, Italian, and Turkish with English mixed in, and CodeMixBench [Yang and Chai, 2025], covering Spanish–English sentiment analysis and TruthfulQA with English mixed into Arabic, Chinese, and Spanish. For each input, we compare routing to the non-English tokenizer, routing to the English tokenizer, and GlotLID-based routing to the most probable tokenizer in the mixture. On `TokSuite` (Table 5), GlotLID routing matches or exceeds the better of the two fixed choices for every language. On CodeMixBench (Table 6), it stays between the two fixed choices, although the differences are small and TruthfulQA accuracy is close to chance for all three.

O.9 *Language-ID routing therefore does not degrade on code-switched inputs, but finer-grained (e.g., span-level) routing may help where the two tokenizers diverge more.*

Table 5: Accuracy (%) of `MoTo` on code-switched `TokSuite` examples (non-English with English mixed in) when routing everything to the non-English tokenizer (Tok. A), to the English tokenizer (Tok. B), or via GlotLID. Best per row in bold.

Lang. A	Tok. A	Tok. B (English)	GlotLID
Chinese	**85**	43	**85**
Farsi	70	33	**72**
Italian	**87**	70	**87**
Turkish	**72**	40	**72**

Table 6: Accuracy (%) of `MoTo` on CodeMixBench when routing everything to the non-English tokenizer (Tok. A), to the English tokenizer (Tok. B), or via GlotLID. Best per row in bold.

Task	Lang. A	Tok. A	Tok. B (English)	GlotLID
Sentiment (3-way)	Spanish	44	**49**	45
TruthfulQA (4–11-way)	Arabic	**22**	**22**	**22**
	Chinese	22	**23**	**23**
	Spanish	21	**24**	22

Without a clear target tokenizer. Superset-Albert, an illustrative baseline rather than our main method, combines fourteen nominally multilingual off-the-shelf tokenizers with no well-defined language-to-tokenizer mapping, so we train and evaluate it with random routing. To estimate the headroom for better routing, Table 7 reports oracle accuracy, where an example counts as correct if any of the fourteen tokenizers yields the correct answer. The gap between random and oracle routing is large on every benchmark, reaching 47.36 points on PAWS-X.

Deconstructing the oracle gap. The oracle in Table 7 is an optimistic noisy upper bound, because choosing the best of fourteen predictions after the fact also rewards chance agreement. PAWS-X shows this most clearly: individual tokenizers are near chance (51.25%), and fourteen independent guesses on a binary task would include a correct answer for $1 - 0.5^{14} > 99.9\%$ of examples, close to the observed 97.3%. To separate signal from noise, we group examples by how many of the fourteen tokenizers answer them correctly. Pooled over all benchmarks, random routing (bootstrapped over 10,000 trials) reaches $38.94 \pm 0.11\%$. $43.15 \pm 0.23\%$ of examples are answered correctly by at least seven tokenizers, and $59.59 \pm 0.22\%$ by at least one. The examples answered correctly by many tokenizers show that routing has real headroom. The rest of the gap up to the oracle includes a large chance component that no realistic router could exploit.

Learned routing. Instead of relying on language ID, the tokenizer could be chosen by a router trained end-to-end with the language model. Since this choice is made before tokenization, the router needs its own representation of the raw input, e.g., a lightweight byte-level encoder like those used in byte-level and latent transformers [Pagnoni et al., 2025]. In preliminary experiments on the synthetic multilingual addition task (§3), such a router proved difficult to train, and we leave learned routing to future work.

Table 7: Percentage of examples answered correctly by at least n of the 14 tokenizers in Superset-Albert. Random routing reaches $38.94 \pm 0.11\%$ accuracy over all benchmarks combined. $\pm$ denotes 95% bootstrap confidence intervals over 10,000 resamples.

Group	Task	N	$\geq$1 tok.	$\geq$7 tok.	All toks.	No tok.
English Reasoning	arc_easy	2376	81.1%	63.9%	12.2%	18.9%
	arc_challenge	1172	55.3%	31.4%	5.2%	44.7%
	hellaswag	10042	50.8%	38.8%	17.3%	49.2%
	piqa	1838	85.9%	70.6%	34.5%	14.1%
	boolq	3270	94.2%	59.3%	5.6%	5.8%
	commonsense_qa	1221	55.1%	18.2%	0.9%	44.9%
			70.4%	47.0%	12.6%	29.6%
XWinograd	en	2325	94.9%	73.0%	26.8%	5.1%
	fr	83	89.2%	60.2%	15.7%	10.8%
	ja	959	98.2%	65.4%	6.7%	1.8%
	pt	263	95.1%	64.6%	15.2%	4.9%
	ru	315	94.6%	61.3%	12.1%	5.4%
	zh	504	98.2%	72.2%	9.3%	1.8%
			95.0%	66.1%	14.3%	5.0%
XCOPA	id	500	85.8%	64.0%	27.8%	14.2%
	it	500	85.0%	59.0%	25.2%	15.0%
	tr	500	83.2%	58.2%	25.4%	16.8%
	vi	500	84.8%	64.2%	28.8%	15.2%
	zh	500	91.8%	59.8%	12.8%	8.2%
			86.1%	61.0%	24.0%	13.9%
XStoryCloze	ar	1511	75.8%	55.1%	26.5%	24.2%
	en	1511	78.8%	64.1%	31.3%	21.2%
	es	1511	76.8%	58.8%	28.9%	23.2%
	id	1511	76.3%	59.2%	29.7%	23.7%
	ru	1511	77.5%	58.6%	28.8%	22.5%
	zh	1511	86.6%	56.3%	15.8%	13.4%
			78.6%	58.7%	26.8%	21.4%
PAWS-X	de	2000	96.8%	57.2%	5.2%	3.1%
	en	2000	97.7%	55.6%	6.3%	2.3%
	es	2000	90.8%	58.6%	14.6%	9.2%
	fr	2000	98.5%	52.9%	2.6%	1.4%
	ja	2000	100.0%	50.6%	0.0%	0.0%
	zh	2000	100.0%	52.4%	0.0%	0.0%
			97.3%	54.6%	4.8%	2.7%
HellaSwag-X	ar	9176	46.9%	31.6%	12.3%	53.1%
	da	9305	49.2%	34.3%	14.7%	50.8%
	de	9368	48.7%	33.6%	14.8%	51.3%
	fr	9338	49.0%	35.1%	15.1%	51.0%
	hu	9119	45.8%	31.7%	14.2%	54.2%
	id	9314	48.4%	33.6%	15.2%	51.6%
	it	9193	48.8%	34.1%	15.1%	51.2%
	nl	9265	49.0%	34.0%	14.2%	51.0%
	pt	9229	49.8%	35.0%	14.9%	50.2%
	ru	9272	48.7%	33.8%	13.4%	51.3%
	vi	9162	46.2%	33.8%	15.7%	53.8%
	es	9374	50.5%	36.4%	15.0%	49.5%
			48.4%	33.9%	14.5%	51.6%
XNLI	ar	2490	51.6%	33.7%	15.7%	48.4%
	zh	2490	70.1%	33.9%	1.2%	29.9%
	de	2490	64.0%	47.3%	13.6%	36.0%
	el	2490	93.2%	40.3%	0.2%	6.8%
	en	2490	76.3%	52.0%	10.0%	23.7%
	es	2490	65.2%	47.6%	6.6%	34.8%
	fr	2490	67.1%	48.4%	9.8%	32.9%
	ru	2490	66.7%	44.5%	15.3%	33.3%
	tr	2490	72.5%	46.2%	4.9%	27.5%
	vi	2490	86.2%	46.8%	0.2%	13.8%
			71.3%	44.1%	7.8%	28.7%

F Merging Per-Language Tokenizers into a Monolithic BPE Tokenizer

Given N language-specific BPE tokenizers $\{T_1, \ldots, T_N\}$, each defined by a corresponding vocabulary V_i, an ordered merge table M_i, and a pretokenizer P_i, we describe a procedure to produce a single monolithic BPE tokenizer T^* with vocabulary V^*, merge table M^*, and pretokenizer P^*.

Vocabulary The superset vocabulary described in Figure 1 serves directly as the vocabulary of the monolithic tokenizer, where the merge priority is determined heuristically below.

Pre-tokenization In general, constituent tokenizers may have divergent pre-tokenization rules, causing the same input string to be split into different pre-tokens across tokenizers This matters because a merge $(a, b) \to ab$ is only valid if a and b can appear as adjacent substrings within a single pre-token; a merge that is reachable under one pre-tokenizer may be unreachable under another. When pre-tokenizers diverge, one must either (1) select a canonical P^* common to all constituent tokenizers and prune all merges in M^* that are unreachable under P^*, propagating pruning through the dependency DAG, or (2) retain per-language pre-tokenization via a language-routing layer.

We adopt option (1) and set P^* to the shared canonical configuration. Pre-tokenization rules help manage the computational complexity of tokenizer training by restricting the space of candidate merges to within-pretoken substrings. Since our method is post-hoc, we are not concerned by this; the relevant cost instead might arise slightly at inference time, where tokenization greedily applies M^* within each pre-token. Because P^* uses a shared, coarser boundary rule than the script-specific pre-tokenizers of the constituent tokenizers, pre-tokens in T^* can span longer substrings, increasing the number of candidate merges considered per pre-token.

In our setting, all constituent tokenizers share identical punctuation, whitespace, number, and byte-level pre-tokenization rules, differing only in script-specific word-boundary patterns. Since no constituent tokenizer learns merges that cross punctuation or whitespace boundaries, the script-specific rules can be dropped without affecting reachability of any merge in M^*.

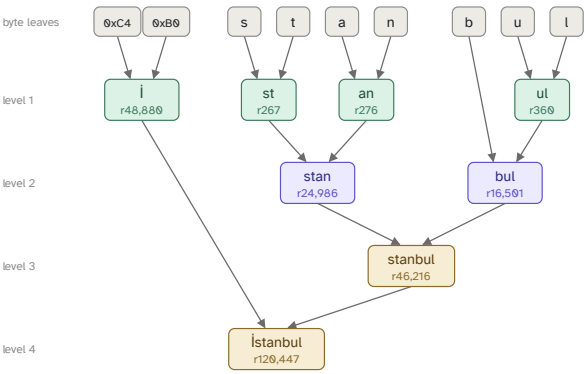

Figure 18: BPE merge tree for İstanbul in the Llama-3.2 tokenizer. Leaves are UTF-8 bytes; each internal node is a merged token annotated with its merge rank (r).

Merge table construction via DAG union. Each merge table M_i induces a dependency DAG G_i whose nodes are tokens. A merge $(a, b) \to ab$ has incoming edges $a \to ab$ and $b \to ab$; base tokens (UTF-8 bytes in most modern tokenizers) have no incoming edges and form the leaves (Figure 18). The union DAG $G^* = \bigcup_i G_i$ takes the union of all nodes and edges, identifying nodes by their token string. Because different tables may build the same token from different pairs, a node in G^* can have more than one incoming merge.

Conflicts arise when two tokenizers T_i and T_j both contain merges m_1 and m_2 but disagree on their relative order, introducing a cycle in G^*. We distinguish two conflict types: *dependency conflicts*, where two tokenizers' dependency chains directly contradict one another (one requires m_1 before m_2 via a chain of structural dependencies, while another requires the reverse), and *ordering conflicts*, where m_1 and m_2 are independent in both DAGs but their relative rank is reversed.

Note that dependency conflicts cannot occur for shared merges: if m_2 is a structural ancestor of m_1 in G_i (i.e. m_1's output string contains m_2's output string as a constituent produced by an earlier merge), this ancestry is a property of the token strings themselves, not of which tokenizer learned it, any tokenizer containing both merges must agree on the direction, since string concatenation is not reversible. All cycles in G^* are therefore necessarily caused by ordering conflicts, which reduce to a rank resolution policy; a resulting cycle may still pass through an uncontested dependency edge alongside one or more conflicting edges, as we observe empirically (Figure 19).

Ordering conflicts are resolved by **majority vote**: for each conflicting pair (m_1, m_2), we retain the ordering agreed upon by the majority of tokenizers containing both merges. For pairs appearing in exactly two tokenizers, majority vote degenerates and we fall back to **min-rank**: we retain the ordering from whichever tokenizer assigns the lower absolute rank to either merge. The resolved G^* is acyclic by construction; we obtain M^* by topological sort, breaking ties by average normalized rank across all tokenizers containing each merge.

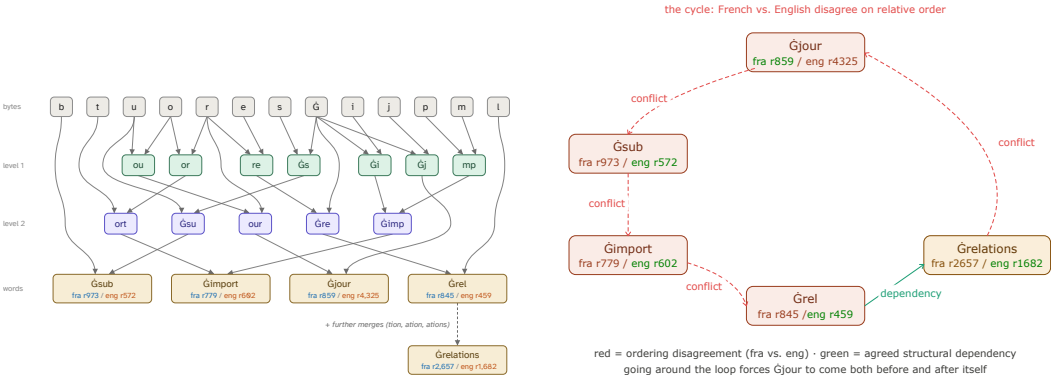

Figure 19: A real ordering-conflict cycle found between the French and English constituent tokenizers. **Left:** dependency tree from byte leaves to the five merges involved in the cycle, with both tokenizers' ranks shown at every node (*fra r / eng r*). **Right:** the cycle isolated. Four edges (red, dashed) are ordering conflicts where French and English disagree on relative rank; one edge (green, solid), `Grel` $\rightarrow$ `Grelations`, is a structural dependency both tokenizers agree on. No linear order of these five merges satisfies both tokenizers simultaneously, motivating the rank resolution policy.

A real example of ordering conflicts Figure 19 (left) shows an ordering conflict found between the French and English constituent tokenizers. The English tokenizer ranks `Gsub`, `Gimport`, and `Grel` before `Gjour`, while the French tokenizer ranks `Gjour` before all three. Since `Grelations` depends structurally on `Grel` (both tokenizers agree that `Grel` must precede `Grelations`, as the latter is built by appending `ations`), the four ordering disagreements together with this one dependency edge close a five-node cycle (Figure 19, right): `Gjour` $\rightarrow$ `Gsub` $\rightarrow$ `Gimport` $\rightarrow$ `Grel` $\rightarrow$ `Grelations` $\rightarrow$ `Gjour`. No single linear order of these five merges is consistent with both tokenizers, illustrating why a direct concatenation or interleaving of merge tables is insufficient and motivating the rank resolution policy.

Vocabulary-composition effects. The DAG-union procedure above resolves conflicts between merges that are *shared* across constituent tokenizers. A distinct failure mode arises from merges that are *not* shared: two tokenizers may segment the same substring in entirely different, non-overlapping ways because their training corpora favored different decompositions, with neither tokenizer's intermediate merges appearing in the other's table at all. This does not produce any cycles, there is no rank to resolve, since the competing merges never co-occur, but it can still yield segmentations that violate morphological boundaries once both decompositions coexist in M^* and greedy BPE is applied.

For example, tokenizing the English word `rescheduled` with the French and English constituent tokenizers yields `res|che|du|le|d` and `res|ched|ul|ed` respectively: neither contains the other's merge (`c`, `he`) vs. (`ch`, `e`) or `c`, `hed`, so the two decompositions are independent rather than conflicting. Critically, the French tokenizer's split reuses two of the most frequent French function-word fragments, `du` ("of the") and `le` ("the"), purely because they are high-rank merges in French and happen to match this English substring by coincidence. Similarly, `unexpected` is split as `une` `x` `pec` `ted` by the French tokenizer, reusing the French feminine indefinite article `une` ("a"), versus the more morphologically sensible `un` `exp` `ected` from the English tokenizer.

Because these cases do not produce cycles, they are invisible to the rank resolution policy and instead emerge from the standard greedy longest-match behavior of BPE once both decompositions are present in M^*. We view this as a limitation of the present method: while DAG-union resolution

guarantees a well-defined, acyclic merge table, it does not guarantee that the resulting segmentations are linguistically coherent for languages whose high-frequency fragments coincide with substrings of unrelated words in other languages. We leave a systematic audit of how frequently such cross-lingual fragment intrusion occurs, and its downstream effect on tokenization quality, to future work.

Constructed monolithic tokenizer (De-MoTo) against MoTo We see that De-MoTo performs worse than MoTo and mBPE alternatives (Figure 20).

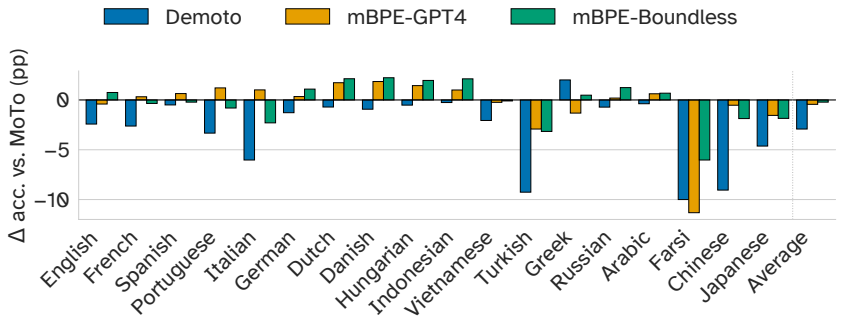

Figure 20: **MoTo against single-vocabulary alternatives.** Per-language accuracy difference (%) from MoTo, averaged over each language's tasks. Bars below zero mean the model is worse than MoTo. Demoto uses MoTo's merged vocabulary as one monolithic tokenizer, so every language is segmented by the same merges but potentially different ordering; mBPE-GPT4 and mBPE-Boundless are multilingual BPE tokenizers trained on the same data mixture, with GPT-4 and Boundless pre-tokenization respectively. The rightmost group is the average over languages.

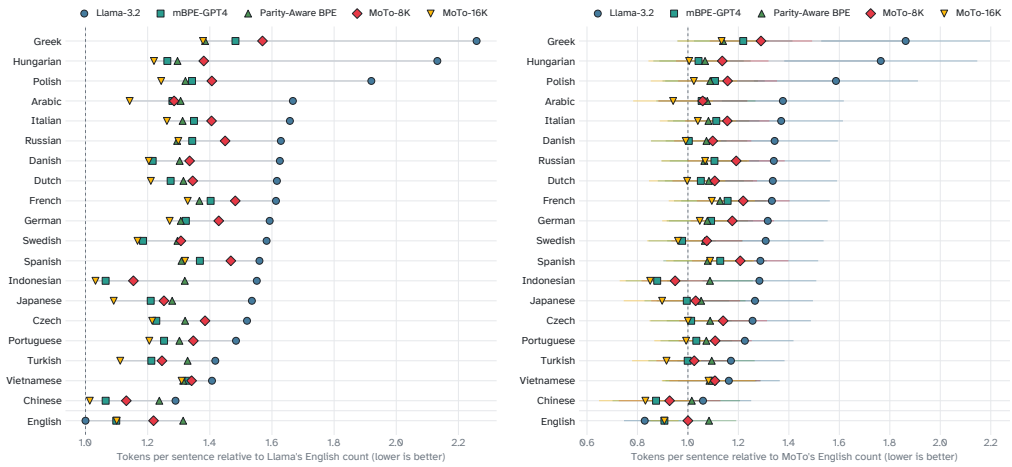

Figure 21: **Relative parity per language** (lower is more fair). Tokens per FLORES+ sentence divided by the English token count of Llama-3.2 (left) or `MoTo` 8K (right), so the dashed line at 1 is the reference tokenizer on English. Each marker is one tokenizer setting, and languages are sorted by Llama's parity with English last. `MoTo` (16K) is the cheapest in most. Full numbers are in Tables 8 and 9.

G Tokenizer Quality & Compression

We report tokenizer quality metrics for all configurations computed on the FLORES+ [Team et al., 2022] dataset using `TokEval` [Meister, 2026] library.

Relative parity. To further examine the relative token cost of each setting, we compare the relative cost of tokenizing the same sentence in that language against English for Llama-3.2 (Figure 21, left and Table 8) and MoTo (Figure 21, right and Table 9). We tokenize the same 1000 parallel FLORES+ devtest sentences in every language and divide each tokenizer's token count by Llama-3.2's count for the English version of that sentence (Table 8), where 1 means "as many tokens as Llama needs for the English sentence"; lower is better parity, where the languages cost similarly across tokenizers. Llama pays the most outside English, from 1.3 on Chinese to 2.3 on Greek (1.60 on average). `MoTo` (8K) needs fewer tokens than Llama on every non-English language and averages 1.35; `MoTo` (16K) averages 1.21, the lowest of all tokenizers in the table, but note that this setting has naturally more capacity than others. Naturally, `MoTo` (8K) needs 22% more tokens than Llama-3 on English (1.22, against 1.10 for mBPE-GPT4), and Parity-Aware BPE needs 31% more.

Cross-language Gini (Table 10). While parity measures cost; the Gini coefficient [Meister, 2026] measures how evenly it is spread. For each tokenizer we take the cost of a language, either tokens per FLORES+ sentence or tokens per UTF-8 byte, and compute the Gini coefficient across languages: 0 means every language costs the same, and larger values mean the cost is concentrated in a few languages. Per sentence, `MoTo` (8K) scores 0.045, the same as the jointly trained mBPE-GPT4 and mBPE-Boundless (0.047, 0.045) and far below Llama (0.082). Parity-Aware BPE is the most even (0.011), as it is trained to equalize parity. Per byte the ranking is similar, except that the byte-level baseline ByT5 scores 0 by construction. We report the per-sentence column and the tokens per sentence next to it, since a low Gini does not reveal how many tokens are spent.

Table 8: Parity against Llama-3 tokenizer: tokens per sentence relative to Llama's English count (mean ± std over sentences; lower is better).

	Baselines				*Per-language*		
Language	**Llama-3.2**	**mBPE-GPT4**	**mBPE-Boundless**	**Parity-Aware BPE**	**MoTo-GPT4**	**MoTo**	**MoTo (16K)**
Arabic	1.67 ± 0.29	1.28 ± 0.22	1.27 ± 0.22	1.31 ± 0.24	1.29 ± 0.23	1.29 ± 0.22	**1.14 ± 0.20**
Czech	1.52 ± 0.26	1.23 ± 0.19	1.23 ± 0.19	1.32 ± 0.22	1.39 ± 0.23	1.38 ± 0.23	**1.21 ± 0.20**
Chinese	1.29 ± 0.26	1.06 ± 0.24	1.09 ± 0.24	1.24 ± 0.27	1.53 ± 0.30	1.13 ± 0.28	**1.01 ± 0.26**
Danish	1.63 ± 0.28	1.22 ± 0.18	1.22 ± 0.18	1.30 ± 0.21	1.33 ± 0.21	1.33 ± 0.21	**1.20 ± 0.18**
German	1.59 ± 0.27	1.32 ± 0.20	1.33 ± 0.20	1.31 ± 0.20	1.42 ± 0.22	1.43 ± 0.22	**1.27 ± 0.19**
Greek	2.26 ± 0.40	1.48 ± 0.26	1.48 ± 0.26	**1.38 ± 0.24**	1.56 ± 0.28	1.57 ± 0.28	**1.38 ± 0.24**
French	1.61 ± 0.26	1.40 ± 0.22	1.39 ± 0.22	1.37 ± 0.22	1.52 ± 0.25	1.48 ± 0.25	**1.33 ± 0.22**
Hungarian	2.13 ± 0.43	1.26 ± 0.22	1.27 ± 0.22	1.30 ± 0.22	1.38 ± 0.24	1.38 ± 0.24	**1.22 ± 0.21**
Indonesian	1.55 ± 0.24	1.06 ± 0.15	1.07 ± 0.15	1.32 ± 0.21	1.15 ± 0.18	1.15 ± 0.18	**1.03 ± 0.15**
Italian	1.66 ± 0.28	1.35 ± 0.21	1.35 ± 0.21	1.31 ± 0.21	1.42 ± 0.23	1.41 ± 0.23	**1.26 ± 0.19**
Japanese	1.54 ± 0.28	1.21 ± 0.22	1.23 ± 0.22	1.28 ± 0.23	2.10 ± 0.40	1.25 ± 0.23	**1.09 ± 0.20**
Dutch	1.62 ± 0.28	1.27 ± 0.20	1.28 ± 0.20	1.32 ± 0.22	1.34 ± 0.23	1.34 ± 0.23	**1.21 ± 0.19**
Polish	1.92 ± 0.37	1.34 ± 0.24	1.34 ± 0.24	1.32 ± 0.24	1.41 ± 0.26	1.41 ± 0.26	**1.24 ± 0.22**
Portuguese	1.48 ± 0.22	1.25 ± 0.17	1.26 ± 0.17	1.30 ± 0.19	1.35 ± 0.20	1.35 ± 0.20	**1.21 ± 0.16**
Russian	1.63 ± 0.29	1.34 ± 0.24	1.35 ± 0.24	**1.29 ± 0.23**	1.48 ± 0.27	1.45 ± 0.26	1.30 ± 0.23
Spanish	1.56 ± 0.27	1.37 ± 0.23	1.37 ± 0.23	**1.31 ± 0.22**	1.47 ± 0.25	1.47 ± 0.25	1.32 ± 0.22
Swedish	1.58 ± 0.26	1.19 ± 0.16	1.19 ± 0.16	1.30 ± 0.19	1.31 ± 0.20	1.31 ± 0.20	**1.17 ± 0.17**
Turkish	1.42 ± 0.25	1.21 ± 0.19	1.21 ± 0.19	1.33 ± 0.22	1.25 ± 0.20	1.25 ± 0.20	**1.11 ± 0.18**
Vietnamese	1.41 ± 0.23	1.33 ± 0.21	1.33 ± 0.21	**1.31 ± 0.21**	1.34 ± 0.21	1.34 ± 0.21	**1.31 ± 0.21**
English	**1.00 ± 0.00**	1.10 ± 0.07	1.10 ± 0.08	1.31 ± 0.13	1.22 ± 0.13	1.22 ± 0.13	1.10 ± 0.08
Average	1.60	1.26	1.27	1.31	1.41	1.35	**1.21**

Table 9: Parity against MoTo: tokens per sentence relative to MoTo's English count (mean ± std over sentences; lower is better).

	Baselines				*Per-language*		
Language	**Llama-3.2**	**mBPE-GPT4**	**mBPE-Boundless**	**Parity-Aware BPE**	**MoTo-GPT4**	**MoTo**	**MoTo (16K)**
Arabic	1.38 ± 0.24	1.05 ± 0.18	1.04 ± 0.17	1.08 ± 0.19	1.06 ± 0.18	1.06 ± 0.18	**0.94 ± 0.16**
Czech	1.26 ± 0.23	1.01 ± 0.16	1.01 ± 0.16	1.09 ± 0.17	1.14 ± 0.17	1.14 ± 0.17	**1.00 ± 0.15**
Chinese	1.06 ± 0.19	0.87 ± 0.17	0.90 ± 0.17	1.02 ± 0.19	1.26 ± 0.24	0.93 ± 0.20	**0.83 ± 0.18**
Danish	1.34 ± 0.25	1.00 ± 0.15	1.00 ± 0.15	1.07 ± 0.16	1.10 ± 0.15	1.10 ± 0.15	**0.99 ± 0.14**
German	1.32 ± 0.24	1.09 ± 0.17	1.09 ± 0.17	1.08 ± 0.16	1.17 ± 0.16	1.18 ± 0.17	**1.05 ± 0.15**
Greek	1.86 ± 0.33	1.22 ± 0.20	1.22 ± 0.19	1.14 ± 0.18	1.29 ± 0.20	1.29 ± 0.20	**1.13 ± 0.17**
French	1.33 ± 0.23	1.16 ± 0.18	1.15 ± 0.18	1.13 ± 0.18	1.25 ± 0.19	1.22 ± 0.18	**1.09 ± 0.17**
Hungarian	1.76 ± 0.38	1.04 ± 0.18	1.05 ± 0.18	1.07 ± 0.18	1.13 ± 0.18	1.14 ± 0.18	**1.00 ± 0.16**
Indonesian	1.28 ± 0.23	0.88 ± 0.12	0.88 ± 0.13	1.09 ± 0.17	0.95 ± 0.13	0.95 ± 0.13	**0.85 ± 0.12**
Italian	1.37 ± 0.24	1.11 ± 0.17	1.11 ± 0.17	1.08 ± 0.16	1.17 ± 0.17	1.16 ± 0.17	**1.04 ± 0.15**
Japanese	1.27 ± 0.23	1.00 ± 0.17	1.01 ± 0.17	1.05 ± 0.18	1.74 ± 0.35	1.03 ± 0.18	**0.90 ± 0.15**
Dutch	1.34 ± 0.25	1.05 ± 0.17	1.05 ± 0.17	1.08 ± 0.17	1.11 ± 0.17	1.11 ± 0.17	**1.00 ± 0.15**
Polish	1.59 ± 0.32	1.11 ± 0.20	1.11 ± 0.20	1.09 ± 0.19	1.16 ± 0.20	1.16 ± 0.20	**1.02 ± 0.17**
Portuguese	1.23 ± 0.19	1.03 ± 0.14	1.04 ± 0.14	1.07 ± 0.15	1.11 ± 0.14	1.11 ± 0.14	**0.99 ± 0.13**
Russian	1.34 ± 0.22	1.11 ± 0.18	1.11 ± 0.18	**1.07 ± 0.17**	1.22 ± 0.20	1.19 ± 0.19	**1.07 ± 0.17**
Spanish	1.29 ± 0.23	1.13 ± 0.18	1.13 ± 0.18	**1.08 ± 0.18**	1.21 ± 0.19	1.21 ± 0.19	1.09 ± 0.17
Swedish	1.31 ± 0.23	0.98 ± 0.13	0.98 ± 0.13	1.07 ± 0.15	1.07 ± 0.14	1.07 ± 0.14	**0.96 ± 0.12**
Turkish	1.17 ± 0.21	1.00 ± 0.16	0.99 ± 0.16	1.09 ± 0.17	1.03 ± 0.15	1.02 ± 0.15	**0.92 ± 0.14**
Vietnamese	1.16 ± 0.20	1.10 ± 0.19	1.10 ± 0.19	1.09 ± 0.19	1.11 ± 0.18	1.11 ± 0.18	**1.08 ± 0.19**
English	**0.83 ± 0.08**	0.91 ± 0.07	0.91 ± 0.07	1.08 ± 0.11	1.00 ± 0.02	1.00 ± 0.00	0.91 ± 0.06
Average	1.32	1.04	1.04	1.08	1.16	1.11	**0.99**

Table 10: Cross-language Gini coefficient of tokenization cost over 20 languages (excluding fa): cost is tokens per FLORES+ sentence (parallel text) or per UTF-8 byte; 0 means every language costs the same. A byte-level tokenizer is 0 per byte by construction, so read the Gini next to the tokens per sentence.

	Gini (sentence)	Gini (byte)	Tokens/sentence	Max/min cost
Llama-3.2	0.082	0.102	42.6	2.23
mBPE-GPT4	0.047	0.071	33.7	1.39
mBPE-Boundless	0.045	0.072	33.8	1.38
mBPE-GPT4 (323K)	0.050	0.073	29.8	1.40
Parity-Aware BPE	**0.011**	0.092	34.9	1.12
ByT5	0.116	**0.000**	163.2	2.38
MoTo-GPT4	0.062	0.103	37.7	1.80
MoTo	0.045	0.080	35.9	1.38
MoTo-16K	0.046	0.078	32.2	1.36
MoTo-16K + Llama	0.049	0.079	32.1	1.37

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: § 2 elaborates the shortcomings of existing tokenization design for different languages, § 2.1 explains the design choices made in `MoTo`, and superset vocabulary construction. § 4 and 3 provide evidence for gains with `MoTo`.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: discussed in § 6 and A. For computational efficiency we further re-index the tokenizers to eliminate additional lookup.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper is an empirical work on language modeling.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Yes, all details for tokenizer training (§ 2.1) and model training and data are disclosed in § C.

   Guidelines:

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

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

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

Answer: [No]

Justification: The models, tokenizers, and code will be open-sourced and published on HuggingFace after acceptance. The size of the models prevent us from releasing them anonymously at this stage.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: We follow the Llama-3.2-1B settings in Grattafiori et al. [2024] and report them in § C.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: While we don't include error bars in our experiments, we compute all metrics based on lm-eval's [Gao et al., 2023] 10,000 bootstrapped estimate per benchmark.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Disclosed in § C.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our work has implications to improve multilingual fairness, which are discussed in the introduction and § A.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Our work has implications that to improve mutlilingual fairness, which are discussed in the introduction and § A.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: As we release base models, immediate risks for misuse are limited. However, we partly address this by training on a higher quality datasets: FineWeb-Edu [Penedo et al., 2024] and FineWeb2-HQ [Messmer et al., 2025], both of which employ filtering to ensure high quality training data.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We use Llama and ByT5 tokenizers within their licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: While we cannot release them at this stage to adhere with double blind policy, all datasets and models will be released with corresponding model and dataset cards on Huggingface.

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

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

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

    Answer: [N/A]

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

Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The method proposed (`MoTo`) is original work of the authors. LLM assistance during the development of this project is declared in § B.

    Guidelines:

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