

Parity-Aware BPE: Improving Cross-lingual Fairness in Tokenization

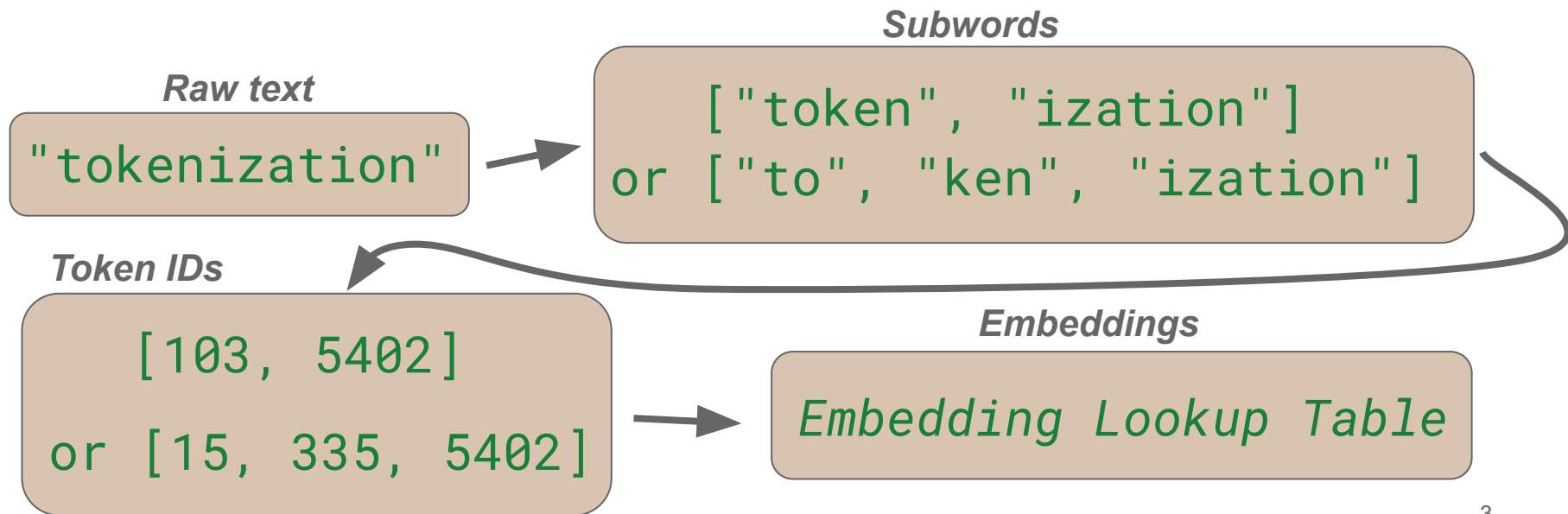
Negar Foroutan*, Clara Meister*, Debjit Paul, Joel Niklaus, Sina Ahmadi,
Antoine Bosselut*, Rico Sennrich*

Motivation + Teaser

- Tokenization = the First Step of most NLP pipelines → Everything that follows (embeddings, modeling, downstream predictions) depends on it
- Tokenization schemes that are optimized for high-resource languages **introduce biases** and incur **unfair costs**
 - High-resource languages get helpful inductive biases
 - vs.
 - Lower-resource languages get more **fragmented + poorer morphological alignment** → **higher API costs** due to longer input/output token sequence lengths + (in some cases) performance hits
- **Solution:** Embed **parity** across languages as an explicit objective in our tokenization algorithms. It works!

What is Tokenization?

- Converts raw text into subword units (tokens); this fixed set of units is then mapped to machine-consumable representations



Tokenization as Compression

The key question: Which spans of text should we merge into 1 token vs. keeping separate?

- Different tokenization algorithms use different principles to choose a vocabulary
- **Compression** is a popular objective
 - **Motivation:** (a) we want to **represent text compactly**: fewer tokens $\rightarrow$ less storage + faster processing. (b) frequent character patterns often align with morphemes, providing semantically coherent tokens
 - **Formula:** optimize the compression rate, defined for a corpus $\mathcal{D}$ as:

$$\text{CR}(\mathcal{D}; \tau) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}|} \sum_{\mathbf{b} \in \mathcal{D}} \frac{|\mathbf{b}|_u}{|\tau(\mathbf{b})|} = \frac{\text{original text length}}{\text{tokenized text length}}$$

Tokenization as Compression

The key question: Which spans of text should we merge into 1 token vs. keeping separate?

- Different subword-level tokenization algorithms use different principles to choose a vocabulary
- Compression is a popular objective
 - Motivation: (a) we want to represent text compactly: fewer tokens → less storage + faster processing, (b) frequent character patterns often align with morphemes,

The Byte-pair Encoding Algorithm greedily maximizes this rate — step by step — by introducing a new token that most reduces total length.

- Formula: optimize the compression rate, defined for a corpus $\mathcal{D}$ as:


$$\text{CR}(\mathcal{D}; \tau) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}|} \sum_{\mathbf{b} \in \mathcal{D}} \frac{|\mathbf{b}|_u}{|\tau(\mathbf{b})|} = \frac{\text{original text length}}{\text{tokenized text length}}$$

Byte Pair Encoding (BPE): The Standard

- BPE is a greedy data compression algorithm adapted for tokenization
- Widely used: GPT, LLaMA

Algorithm steps: ***Learning** the tokenization scheme*

- Start from characters (or bytes) and identify the pair of symbols v, v' that co
- *Merge* them: Replace (v, v') with a new symbol v'' everywhere in the corpus and recompute frequency statistics
- Repeat until desired vocabulary size is reached
- Return, the ordered list of merges, which defines how future text is tokenized

Byte Pair Encoding (BPE): The Standard

- BPE is a greedy data compression algorithm adapted for tokenization
- Widely used: GPT, LLaMA
- Issue: frequency-based optimization favors high-resource languages
 - When learning our list of merges $\mathbf{m}$, goal is overall corpus compression

$$\mathbf{m}^* = \max_{\mathbf{m}} \text{CR}(\mathcal{D}, \tau_{\mathbf{m}}) \quad \text{s.t.} \quad |\mathbf{m}| = K$$

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


Byte Pair Encoding (BPE): The Standard

- BPE is a greedy data compression algorithm adapted for tokenization
- Widely used: GPT, LLaMA
- Issue: frequency-based optimization favors high-resource languages
 - When learning our list of merges $\mathbf{m}$, goal is overall corpus compression
 - In multilingual corpora, tokens that compress high-resource languages will be prioritized. Resulting per-language compressions can differ hugely. I.e., for ℓ, ℓ' :

$$\text{CR}(\mathcal{D}_{\ell}; \tau) \neq \text{CR}(\mathcal{D}_{\ell'}; \tau)$$

Byte Pair Encoding (BPE): The Standard

- BPE is a greedy data compression algorithm adapted for tokenization
- Widely used: GPT, LLaMA
- Issue: frequency-based optimization favors high-resource languages
 - When learning our list of merges **m**, goal is overall corpus compression
 - In multilingual corpora, tokens that compress high-resource languages will be prioritized. Resulting per-language compressions can differ hugely

Example: English vs. Low-Resource Language Tokenization Length

Sentence	Language	GPT Tokenization	Tokens per Word
"I love learning languages."	English	["I", " love", " learning", " languages", "."]	1.0
“මට භාෂා ඉගෙන ගන්න කැමතියි.” (Maṭa bhāṣā igen ganna kœmatiyi, Sinhala)	Sinhala	["ම", "ට", " භ", "ා", "ෂ", "ා", " ඉ", "ගෙ", "න", " ග", "න්න", " ක", "ැ", "ම", "තියි", "."]	~3.5

Enter: Parity-Aware BPE

- Key idea: Replace global optimization with min-max fairness criterion

$$\mathbf{m}^* = \max_{\mathbf{m}} \text{CR}(\mathcal{D}, \tau_{\mathbf{m}}) \quad \text{s.t.} \quad |\mathbf{m}| = K$$



$$\mathbf{m}^* = \max_{\mathbf{m}} \min_{\ell, \tau_{\mathbf{m}}} \text{CR}(\mathcal{D}_{\ell}, \tau_{\mathbf{m}}) \quad \text{s.t.} \quad |\mathbf{m}| = K$$

- Rather than maximizing compression at each step globally, we maximize compression for the worst-off language

Enter: Parity-Aware BPE

Standard BPE Learning Algorithm

Algorithm steps: ***Learning** the tokenization scheme*

- Start from characters (or bytes) and **identify the pair of symbols v , v' that co-occur most frequently in the corpus**
- Replace (v, v') with a new symbol v'' everywhere in the corpus and recompute frequency statistics
- Repeat until desired vocabulary size is reached
- Return $\mathbf{m}$, the ordered list of merges, which defines how future text is tokenized

Enter: Parity-Aware BPE

Only merge decision rule
changes—rest stays the same!

Parity-aware BPE Learning Algorithm

Algorithm steps: ***Learning** the tokenization scheme*

- Start from characters (or bytes) and identify the language with the worst compression rate
 - Identify the pair of symbols v, v' that co-occur most frequently in the portion of the corpus corresponding to that language
- Replace (v, v') with a new symbol v'' everywhere in the corpus and recompute frequency statistics
- Repeat until desired vocabulary size is reached
- Return $\mathbf{m}$, the ordered list of merges, which defines how future text is tokenized

Parity-Aware BPE: Practical Considerations

Problem: Trading off global efficiency and fairness

Depending on developers' goals, they may wish to vary the degree to which they prioritize global compression vs. cross-lingual compression equality

Solution: *Hybrid Parity-Aware BPE*

- Start with the **global compression objective** (Classical BPE) for K merges.
- Switch to the **fairness objective** for J merges.
- Lets developers **balance fairness and efficiency** depending on available data and priorities.

Parity-Aware BPE: Practical Considerations

Problem: No access to parallel corpora

To find the worst-off language, the algorithm compares compression across languages. The default uses a parallel corpus to compute this, but we may not have access to such a corpus.

Solution: *Ratios variant*

Use a fixed per-language ratio to compare compression across languages, rather than measuring it on a parallel corpus. The ratios can be:

- computed once from an existing parallel corpus, or
- estimated from prior knowledge about the language. The fairness objective then runs without parallel data at training time.

Experiments: Setup

Datasets and Models

Tokenizer training: multilingual C4 (**mC4**) corpus and FineWeb2

Development set: FLORES+ dev corpus (NLLB Team, 2024) for compression tracking.

- *No-dev* variant uses mC4 directly (normalized by bytes)

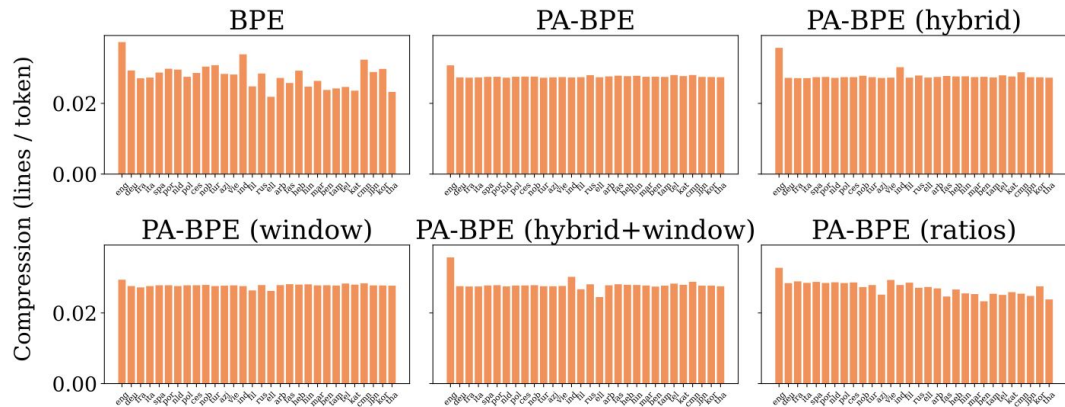
Language sets:

- **30-lang** and **60-lang** groups, unbalanced proportions (reflecting realistic data distribution)
- Languages categorized by resource level: High (>1M examples), Medium (0.5–1M), Low (<0.5M).

Model & Hyperparameters

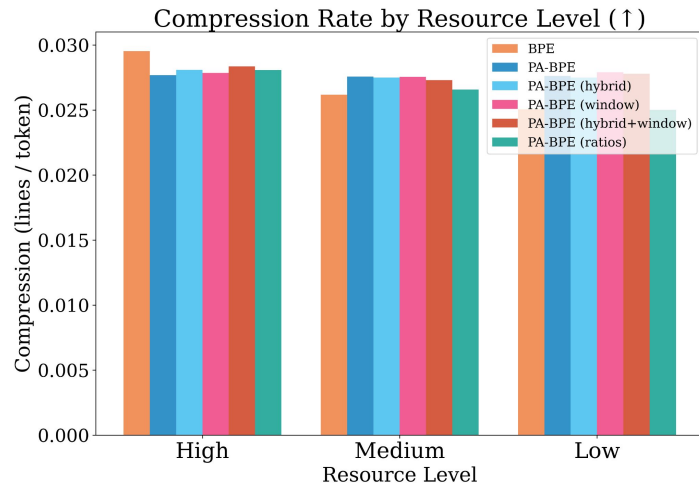
- **Tokenizer vocab sizes:** 128k and 256k.
- **Baseline:** Classical BPE.

Experiments: Intrinsic Results



- Gini ↓ from 0.064 → 0.007
- Vocabulary utilization ↑ (esp. low-resource)
- Global compression essentially unchanged

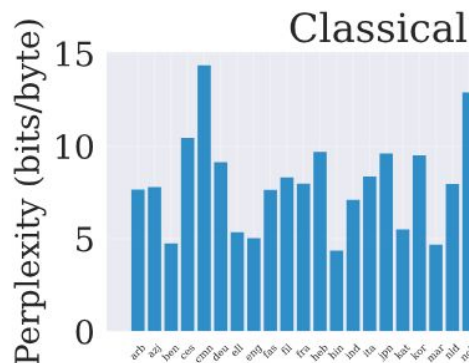
Fairness gains without efficiency loss!



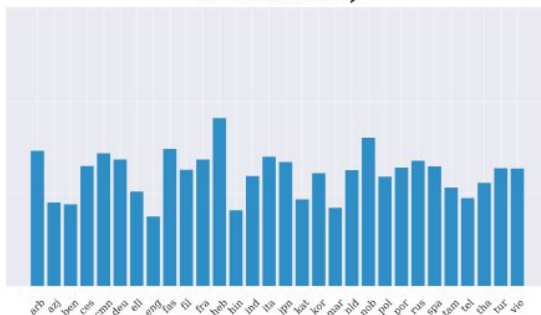
Language	Classical BPE	Parity-aware (hybrid)	Parity-aware (window+hybrid)	Random
Arabic	38.19 $\pm$ 2.90	39.04 $\pm$ 2.89	38.84 $\pm$ 2.90	32.00
Bengali	24.95 $\pm$ 3.09	23.54 $\pm$ 2.98	23.91 $\pm$ 3.01	25.00
German	32.92 $\pm$ 3.14	34.78 $\pm$ 3.66	36.82 $\pm$ 4.04	30.62
Greek	41.95 $\pm$ 3.18	42.55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

Experiments: Extrinsic Results

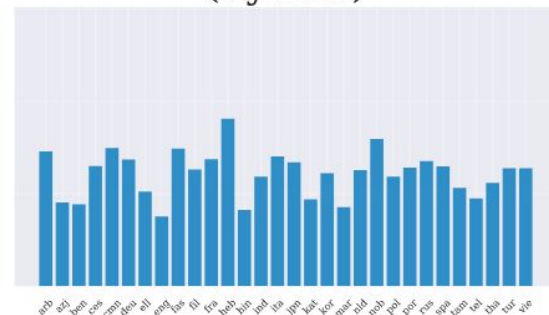
Parity-aware BPE matches
Classical BPE!



Parity-aware
(hybrid +
window)



Parity-aware
(hybrid)



Next Steps

- Can this extend to UnigramLM, WordPiece?
- What about fairness in code/speech?
- Compression isn't the only fairness measure
- Path to wider adoption in LLM training?