

# Revisiting the Effectiveness of Word-level Tokenization in Language Model Pretraining

Takahiro Shimizu<sup>1</sup> Tianqi Wang<sup>1</sup> Jun Suzuki<sup>1,2,3</sup>

<sup>1</sup>Tohoku University <sup>2</sup>RIKEN <sup>3</sup>NII LLMC

is-failab-research@grp.tohoku.ac.jp

## Abstract

Subword tokenization is widely used in modern large language models for open-vocabulary coverage and token efficiency. We revisit word-level tokenization by comparing it with SentencePiece Unigram under matched byte fallback settings, allowing both tokenizers to represent out-of-vocabulary units without unknown tokens. Using language-appropriate word-like segmentation, we evaluate models pretrained from scratch across six model configurations in English and, using a common model configuration, across four languages (English, French, Chinese, and Japanese), with vocabulary sizes of 10k, 50k, and 100k. Although subword tokenization is generally more token-efficient, raw accuracy varies across model configurations, languages, and vocabulary sizes, with word-level tokenization remaining competitive across the evaluated settings. Word-level tokenization also tends to achieve higher character-length-normalized accuracy, although this complementary metric does not isolate the effects of tokenization on model scoring or establish a universal advantage in language modeling quality. These results show that word-level tokenization can remain competitive with subword tokenization when open-vocabulary coverage and language-appropriate segmentation are provided. Our code and tokenizers are publicly available.<sup>1</sup>

## 1 Introduction

Input text to large language models (LLMs) is typically converted into token sequences by a predefined tokenizer. Most modern LLMs use subword tokenization methods such as byte-pair encoding, unigram language-model tokenization, and WordPiece (Sennrich et al., 2016; Kudo, 2018; Kudo and Richardson, 2018; Devlin et al., 2019). These methods provide open-vocabulary coverage with

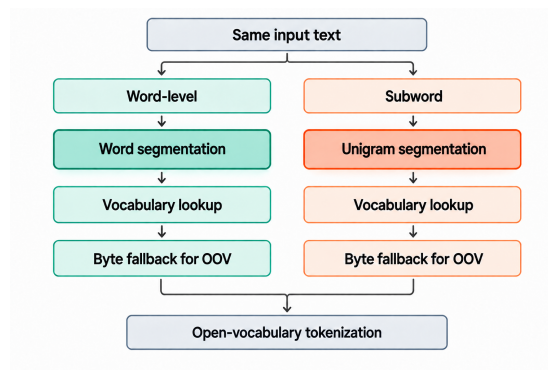


Figure 1: Overview of word-level and SentencePiece Unigram tokenization. Both methods support byte fallback for open-vocabulary coverage, while differing in their segmentation strategies.

relatively compact vocabularies, but can split words into multiple pieces and obscure word-level units.

We revisit word-level tokenization, which was widely used in earlier neural language processing (Jean et al., 2015; Luong et al., 2015). Conventional word-level tokenizers often mapped out-of-vocabulary (OOV) words to an unknown token, discarding their surface forms. In contrast, we implement word-level tokenization with byte fallback, representing OOV units as UTF-8 byte sequences rather than unknown tokens. Here, an *OOV unit* denotes a segmented input unit absent from the word-level entry vocabulary. Because word boundaries are language-dependent, we use rule-based segmentation for English and French, jieba for Chinese, and MeCab for Japanese. Figure 1 illustrates the difference in segmentation between word-level and subword tokenization, while highlighting their shared byte fallback capability.

We compare word-level and SentencePiece Unigram tokenization under controlled pretraining settings. We evaluate six model configurations from four architecture families in English and conduct multilingual experiments in English, French, Chinese, and Japanese using Llama3.2 1B, with target

<sup>1</sup>[tohoku-nlp/word-level-tokenization](https://github.com/tohoku-nlp/word-level-tokenization)

vocabulary sizes of 10k, 50k, and 100k. Within each paired comparison, we hold the model configuration, pretraining corpus, optimization settings, context length, random seed, and emitted-token budget fixed while varying the tokenizer. Because tokenizers differ in sequence length, however, a fixed emitted-token budget can result in different amounts of raw-text exposure.

Our results show that SentencePiece Unigram is generally more token-efficient, but this advantage does not consistently translate into higher downstream accuracy. Raw accuracy varies across model configurations, languages, and vocabulary sizes, while word-level tokenization remains competitive across the evaluated settings. Word-level tokenization more consistently achieves higher character-length-normalized accuracy, although this complementary metric does not isolate the effects of tokenization on model scoring or establish a universal advantage in language modeling quality.

Our contributions are threefold. First, we compare word-level and SentencePiece Unigram tokenization under matched byte fallback availability and controlled pretraining settings across multiple model configurations, vocabulary sizes, and languages. Second, we show that word-level tokenization remains competitive when language-appropriate word-like segmentation and open-vocabulary coverage are provided. Third, we quantify the trade-off between token efficiency and downstream performance, showing that greater token efficiency does not consistently correspond to higher downstream accuracy.

## 2 Related Work

Tokenization is a fundamental component of neural NLP systems, affecting vocabulary coverage, sequence length, computational efficiency, multilingual modeling, and representation (Mielke et al., 2021). Prior work has documented the transition from word-level models to character-, subword-, and byte-level open-vocabulary modeling (Mielke et al., 2021), alongside developments in neural machine translation and pretrained language models (Jean et al., 2015; Sennrich et al., 2016; Devlin et al., 2019; Raffel et al., 2020).

Before subword tokenization became dominant, neural NLP systems commonly used word-level tokenization (Jean et al., 2015; Luong et al., 2015). Such systems often mapped out-of-vocabulary words to unknown tokens, creating a trade-off be-

tween vocabulary size and OOV frequency (Jean et al., 2015; Sennrich et al., 2016). Subword methods instead decompose rare words into smaller units while retaining frequent words as single tokens (Sennrich et al., 2016), enabling compact vocabularies with broad coverage in models such as BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020). Shorter sequences can improve computational efficiency and sometimes downstream performance (Gall , 2019), while vocabulary construction and optimization are themselves important design choices (Xu et al., 2021; Song et al., 2021). However, tokenization is not solely a compression problem: segmentation choices can affect model behavior, and shorter sequences do not necessarily yield better downstream performance (Schmidt et al., 2024). Vocabulary size can also affect LLM performance (Takase et al., 2025).

SentencePiece provides language-independent subword tokenization directly from raw text and supports byte fallback (Kudo and Richardson, 2018). It is widely used in multilingual settings, including languages for which whitespace does not provide suitable segmentation boundaries (Kudo and Richardson, 2018; Raffel et al., 2020). We use SentencePiece Unigram as the primary subword baseline and additionally evaluate SentencePiece BPE, with byte fallback enabled for both subword and word-level tokenizers.

Prior work has also studied segmentation diversity and robustness through methods such as Subword Regularization (Kudo, 2018) and BPE-Dropout (Provilkov et al., 2020), as well as fragmentation, sequence efficiency, and representation effects (Chai et al., 2024; Gall , 2019; Mielke et al., 2021). These studies highlight tokenization as a modeling choice rather than only a compression mechanism.

Other work has reconsidered the boundary between tokenization and model architecture (Thawani et al., 2023; Chai et al., 2024; Feher et al., 2025). Examples include word-boundary pooling (Thawani et al., 2023), latent or dynamically constructed subword representations (Tay et al., 2022), dynamic tokenization (Feher et al., 2025), tokenizer-free byte- or character-level models such as ByT5, CANINE, MEGABYTE, and Charformer (Xue et al., 2022; Clark et al., 2022; Yu et al., 2023; Tay et al., 2022), and patch-based byte-level approaches such as Byte Latent Transformer (Pagnoni et al., 2025). Unlike these approaches, our study retains explicit tokenization

and a conventional token-based language-modeling setup.

Tokenization behavior also depends on linguistic and morphological properties (Toraman et al., 2023; Mielke et al., 2021). Morphologically rich languages can exhibit different tokenization behavior and sequence lengths from English (Toraman et al., 2023), and tokenizer behavior is sensitive to language variation (Wegmann et al., 2025). Because whitespace is not a universal proxy for word boundaries, morphology-aware and factorized subword methods have been proposed (Samuel and Øvreliid, 2023), while vocabulary allocation and overlap have been studied in multilingual language modeling (Limisiewicz et al., 2023). Ali et al. (2024) examine tokenizer choice in LLM training, and Brahma et al. (2026) study multilingual tokenization for Indian languages. In contrast, we compare externally supplied word-like segmentation with learned subword tokenization, using rule-based segmentation for English and French and language-specific segmenters for Chinese and Japanese.

Finally, byte fallback allows strings outside a fixed vocabulary to be represented without an unknown token. Our word-level tokenizer represents an OOV segmented unit by its UTF-8 bytes, separating word-like segmentation from the conventional unknown-token limitation. Unlike tokenizer-free models (Xue et al., 2022; Clark et al., 2022; Yu et al., 2023) or methods that modify the interaction between tokenization and model architecture (Tay et al., 2022; Feher et al., 2025), we directly compare word-level and subword tokenization under matched pretraining settings, with byte fallback available in both conditions.

### 3 Word-Level Tokenization

We revisit word-level tokenization under an open-vocabulary setting in which out-of-vocabulary (OOV) units are represented without being mapped to an unknown token. Because “word” does not correspond to a single language-independent segmentation rule, we construct language-aware word-level tokenizers using segmentation procedures appropriate for each language.

For English, we use rule-based segmentation over ASCII alphabetic sequences, individual digits, other non-whitespace characters, and whitespace. For French, we apply Unicode NFC normalization, apostrophe normalization, lowercasing, and whitespace normalization before rule-based segmenta-

tion that handles accented alphabetic sequences and common apostrophe-based clitic boundaries. For Chinese, we use jieba (fxsjy, n.d.) to segment non-whitespace text. For Japanese, we use MeCab (Kudo et al., 2004) through fugashi (McCann, 2020) and retain the surface forms of the resulting morphological units. For Chinese and Japanese, whitespace is handled separately from language-specific segmentation. These procedures are not intended to provide a unique linguistic definition of a word; rather, they provide reproducible, externally supplied word-like boundaries for comparing lexical segmentation with statistically learned subword segmentation.

#### 3.1 Vocabulary Construction

For each language, we apply the corresponding segmentation procedure to the tokenizer-training corpus and rank the resulting units by frequency. The word-level vocabulary includes special tokens, a dedicated whitespace token, and 256 byte-fallback tokens. The remaining vocabulary entries are assigned to the most frequent segmented units. We refer to the vocabulary settings by their nominal target sizes of 10k, 50k, and 100k.

At tokenization time, the input is first processed and divided into word-like units according to the language-specific procedure. Each resulting unit is then looked up independently in the fixed vocabulary using the corresponding language-specific normalization rules. An in-vocabulary unit is emitted as a single token, whereas an OOV unit is converted into a sequence of UTF-8 bytes, with each byte represented by one of the 256 byte-fallback tokens. For English, non-whitespace units are lowercased for vocabulary lookup, but OOV units are byte-encoded using their forms before this lookup-time lowercasing. For French, normalization is applied before segmentation, so byte fallback operates on the normalized units. For Chinese and Japanese, byte fallback operates on the units returned by their respective segmenters. Segmentation therefore determines the candidate lexical units, while vocabulary membership determines whether each unit is represented directly or through byte fallback.

This design separates lexical segmentation from unknown-token handling. Conventional word-level tokenizers can discard the surface form of an unseen unit by replacing it with an unknown token, whereas byte fallback retains a byte-level representation of the corresponding unit. The resulting tokenizer therefore provides open-vocabulary coverage

while preserving frequent segmented units as single tokens.

Detailed normalization rules, segmentation patterns, language-specific implementation choices, and available tokenizer-training corpus specifications are provided in Appendix A.

### 3.2 Subword Baseline

Our primary subword baseline is SentencePiece Unigram. For each language, the Unigram tokenizer is trained on the corresponding language-specific tokenizer-training data, with target vocabulary-size settings of 10k, 50k, and 100k. Both tokenizer conditions support byte fallback.

Unlike the word-level tokenizer, which receives externally supplied word-like boundaries, SentencePiece Unigram learns its segmentation statistically from the tokenizer-training corpus. Consequently, a word-like unit identified by the external segmentation procedure need not correspond to a single Unigram token and may instead be represented by multiple subword pieces. The comparison therefore contrasts externally specified word-like segmentation with statistically learned subword segmentation while retaining open-vocabulary coverage in both conditions.

For French, the word-level and Unigram tokenizers are constructed using the same corpus of five million normalized documents. The available tokenizer-training corpus specifications, including the distinction between tokenizer training and model pretraining in the Chinese experiments, are detailed in Appendix A.5.

As an additional robustness check, we compare the word-level tokenizer with SentencePiece BPE under the same multilingual Llama3.2 1B setup. These results are reported separately in Appendix C.

## 4 Experimental Setup

We evaluate word-level tokenization along two axes: model architecture and scale, and language. First, we compare tokenization strategies across multiple model configurations in English. Second, we compare them across languages using language-appropriate word-like segmentation. The main experiments compare word-level tokenization with SentencePiece Unigram, while additional comparisons with SentencePiece BPE are reported in Appendix C.

### 4.1 Models and Pretraining

All models are initialized from scratch. Model names such as SmolLM2 (Ben Allal et al., 2025), Gemma3 (Gemma Team, 2025), Qwen3 (Qwen Team, 2025), and Llama3.2 (Llama Team, 2024; Meta AI, 2024) refer only to their architecture configurations; no pretrained weights are used.

For the English experiments, we evaluate six model configurations from four architecture families: SmolLM2 135M, SmolLM2 360M, SmolLM2 1.7B, Gemma3 270M, Qwen3 0.6B, and Llama3.2 1B. For each configuration, we compare word-level tokenization with SentencePiece Unigram at target vocabulary-size settings of 10k, 50k, and 100k.

The English pretraining budgets scale approximately with model size. SmolLM2 135M, 360M, and 1.7B are trained for 2.7B, 7.2B, and 34B emitted tokens, respectively. Gemma3 270M is trained for 5.4B tokens, Qwen3 0.6B for 12B tokens, and Llama3.2 1B for 20B tokens.

For the multilingual experiments, we use the Llama3.2 1B architecture for English, French, Chinese, and Japanese. All models are trained for 20B emitted tokens with a context length of 1024. For each language, we evaluate target vocabulary-size settings of 10k, 50k, and 100k and compare word-level tokenization with SentencePiece Unigram. The additional BPE comparisons use the same multilingual Llama3.2 1B setup.

Within each paired comparison, the tokenization condition is the primary experimental component intentionally varied. We keep the model architecture, pretraining corpus, optimizer settings, context length, random seed, and emitted-token budget fixed. Vocabulary size is matched at the level of the target vocabulary-size setting, while the resulting token sequences and raw-text exposure are allowed to differ as consequences of the tokenization strategy.

Accordingly, the emitted-token budget, rather than the amount of raw text, is matched between tokenizer conditions. Because different tokenizers can encode the same raw text using different numbers of tokens, paired models may process different numbers of documents, characters, and bytes before reaching the common token budget. Our comparisons therefore control the training budget in terms of emitted tokens rather than the amount of underlying text. We quantify the resulting differences in raw-text exposure directly in Section 6 and account for them when interpreting downstream

performance.

Detailed optimization and training settings are provided in Appendix B.

## 4.2 Pretraining Data

English models are pretrained on FineWeb-Edu-Dedup (Ben Allal et al., 2024; Penedo et al., 2024). French and Chinese models are pretrained on the corresponding language subsets of FineWeb2 (Penedo et al., 2025). Japanese models are pretrained on FineWeb2-Edu Japanese (Penedo et al., 2025; Tatenno, 2025). Within each language, the word-level and subword conditions use the same model-pretraining corpus.

The tokenizer-training corpus need not be identical to the model-pretraining corpus. In the Chinese experiments, the documented SentencePiece tokenizer-training procedure uses data derived from `opencsg/chinese-fineweb-edu-v2`, whereas the models are pretrained on the `cmn_Hani` subset of FineWeb2. Further details of the available tokenizer-training corpus specifications are provided in Appendix A.5.

The word-level tokenizers apply language-specific normalization and segmentation procedures. For English, non-whitespace units are lowercased for vocabulary lookup, whereas OOV units are byte-encoded using their forms before this lookup-time lowercasing. For French, Unicode NFC normalization, apostrophe normalization, lowercasing, and whitespace normalization are applied before segmentation. For Chinese and Japanese, whitespace and non-whitespace spans are handled separately, with jieba and MeCab, respectively, applied to non-whitespace text. The language-specific word-level segmentation procedures are described in Section 3, with implementation details in Appendix A.

## 4.3 Evaluation Tasks

We evaluate all models in the zero-shot setting using `lm-evaluation-harness` (Biderman et al., 2024). Because equivalent benchmark suites are not available across all four languages, we use six language-appropriate tasks for each language.

For English, we evaluate ARC Easy, ARC Challenge (Clark et al., 2018), SWAG (Zellers et al., 2018), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), and OpenBookQA (Mihaylov et al., 2018). For French, we use ARC Easy, ARC Challenge, PIQA, HellaSwag, French Grammar, and French Vocabulary (Faysse et al., 2025).

For Chinese, we evaluate Belebele (Bandarkar et al., 2024), ARC, PIQA, PAWS-X (Yang et al., 2019), XCOPA (Ponti et al., 2020), and XWinograd (Tikhonov and Ryabinin, 2021). For Japanese, we use Belebele, JARC (Clark et al., 2018; weblab-GENIAC, n.d.), PIQA, PAWS-X, Japanese COPA (Roemmele et al., 2011; nlp-titech, n.d.), and XWinograd.

These task sets are used to compare tokenizer conditions within each language, rather than to compare absolute performance across languages. Accordingly, language-level averages are macro-averaged over the six evaluation tasks for each language.

## 4.4 Evaluation Metrics

We report both accuracy (`acc`) and character-length-normalized accuracy (`acc_norm`) as implemented in `lm-evaluation-harness` (Biderman et al., 2024). Let  $s$  denote a candidate answer string, and let  $c^{(\tau)} = (c_1, \dots, c_{T_\tau})$  be its tokenized representation under tokenizer  $\tau$ . For a prompt  $x$ , raw multiple-choice scoring assigns the candidate the total conditional log-likelihood

$$\text{score}_{\text{raw}}(s; \tau) = \sum_{t=1}^{T_\tau} \log p_\tau(c_t \mid x, c_{<t}). \quad (1)$$

Normalized scoring divides the raw score by the number of characters in the candidate answer string:

$$\text{score}_{\text{norm}}(s; \tau) = \frac{\text{score}_{\text{raw}}(s; \tau)}{L_{\text{char}}(s)}. \quad (2)$$

Here,  $L_{\text{char}}(s)$  is the character length of the candidate string, as computed by the evaluation harness. This denominator is not the number of tokens produced by tokenizer  $\tau$ .

For each evaluation instance, let  $\mathcal{C}(x)$  denote the set of candidate answer strings for prompt  $x$ . The predicted answer is selected by maximizing the corresponding candidate score:

$$\hat{s} = \arg \max_{s \in \mathcal{C}(x)} \text{score}(s; \tau), \quad (3)$$

where `score` is either `scoreraw` or `scorenorm`. Accuracy is the proportion of evaluation instances for which the selected candidate matches the gold answer.

For a given candidate string, the character-length denominator is the same across tokenizers,

whereas its token sequence and conditional log-likelihood can differ. Consequently, `acc_norm` controls for candidate character length but does not isolate tokenization effects from other differences in model scoring. We treat raw accuracy as the primary downstream metric and use character-length-normalized accuracy as a complementary measure.

For compactness, the main result tables report the difference between word-level and Unigram performance. For metric  $m \in \{\text{acc}, \text{acc\_norm}\}$ , we define

$$\Delta m = m_{\text{word}} - m_{\text{Unigram}}. \quad (4)$$

Positive values favor word-level tokenization, whereas negative values favor SentencePiece Unigram. The BPE results in the appendix use the same convention, replacing  $m_{\text{Unigram}}$  with  $m_{\text{BPE}}$ .

## 5 Experimental Results

### 5.1 Generalization in English

Table 1 compares word-level tokenization with SentencePiece Unigram across six English model configurations and three vocabulary sizes. The raw-accuracy results are mixed rather than uniformly favoring either tokenizer. At 10k, the average raw-accuracy difference is negative for all six model configurations, whereas at 50k and 100k it is positive in five and four of the six configurations, respectively.

Aggregated over all 18 architecture–vocabulary settings, the mean raw-accuracy difference is approximately zero. Thus, across the evaluated English architectures and scales, neither tokenizer shows a consistent advantage in raw accuracy.

Character-length-normalized accuracy shows a more consistent pattern: the average  $\Delta \text{acc\_norm}$  is positive in all 18 architecture–vocabulary settings, with a mean difference of approximately +0.023. Because the denominator is the character length of the candidate string rather than its tokenizer-specific token count, this result should not be interpreted as an advantage arising from token-count normalization. The metric provides a complementary view of downstream performance, while the tokenizer-dependent likelihoods remain part of the comparison.

The differences also remain task-dependent. For example, ARC Challenge frequently favors Unigram in raw accuracy, whereas OpenBookQA shows comparatively large positive differences in normalized accuracy. Overall, word-level tokenization remains competitive across the evaluated

model configurations, while relative performance varies across tasks and vocabulary sizes.

### 5.2 Multilingual Generalization

Table 2 extends the comparison to English, French, Chinese, and Japanese using the Llama3.2 1B architecture. The average raw-accuracy difference is positive in 11 of the 12 language–vocabulary settings, with the only negative average occurring for English at 10k (−0.0045). The average normalized accuracy difference is positive in all 12 settings.

The largest average raw-accuracy differences occur in French, driven in substantial part by the French Grammar and French Vocabulary tasks. However, ARC Easy, ARC Challenge, and PIQA include settings in which Unigram performs better, indicating substantial task-level variation.

Chinese and Japanese provide complementary settings in which word boundaries are not generally marked by whitespace. Using jieba for Chinese and MeCab for Japanese, the average raw-accuracy difference is positive at all three vocabulary sizes in both languages. Together with the English and French results, these findings show that the competitiveness of word-level tokenization is not limited to English and is observed across the evaluated languages when language-specific word-like segmentation is used.

### 5.3 Effect of Vocabulary Size

Vocabulary size affects the tokenizer comparison, but the pattern is not uniform across languages. In the English architecture experiments, the average raw-accuracy difference is negative for all six model configurations at 10k, whereas positive differences become more common at 50k and 100k (Table 1). In contrast, French, Chinese, and Japanese show positive average differences at all three vocabulary sizes (Table 2). The differences do not increase monotonically with vocabulary size, indicating no simple monotonic relationship between word-level vocabulary size and downstream accuracy.

For the word-level tokenizer, increasing vocabulary size allows more frequent segmented units to be represented directly rather than through byte fallback. As analyzed in Section 6, this reduces byte-fallback usage and increases the amount of raw text processed under the fixed emitted-token budget. However, these efficiency changes do not establish a causal explanation for the observed downstream-accuracy differences. Vocabulary size also affects the model’s embedding and output layers and may

| Task                                       | SmolLM2 135M |         |         | SmolLM2 360M |         |         | SmolLM2 1.7B |         |         |
|--------------------------------------------|--------------|---------|---------|--------------|---------|---------|--------------|---------|---------|
|                                            | 10k          | 50k     | 100k    | 10k          | 50k     | 100k    | 10k          | 50k     | 100k    |
| Accuracy ( $\Delta acc$ )                  |              |         |         |              |         |         |              |         |         |
| ARC Easy                                   | +0.0029      | +0.0012 | +0.0088 | +0.0152      | +0.0033 | -0.0034 | +0.0126      | +0.0168 | +0.0164 |
| ARC Challenge                              | -0.0187      | -0.0094 | +0.0103 | -0.0281      | -0.0042 | -0.0145 | -0.0221      | -0.0350 | -0.0282 |
| SWAG                                       | +0.0089      | +0.0087 | +0.0059 | +0.0056      | +0.0092 | +0.0071 | +0.0041      | +0.0055 | +0.0083 |
| PIQA                                       | -0.0185      | -0.0065 | +0.0060 | -0.0169      | +0.0125 | +0.0065 | +0.0038      | -0.0081 | +0.0103 |
| HellaSwag                                  | +0.0018      | +0.0126 | +0.0100 | +0.0058      | +0.0118 | +0.0120 | +0.0082      | +0.0097 | +0.0074 |
| OpenBookQA                                 | -0.0560      | +0.0080 | +0.0180 | -0.0540      | +0.0100 | -0.0280 | -0.0380      | -0.0020 | +0.0400 |
| Avg.                                       | -0.0133      | +0.0024 | +0.0098 | -0.0121      | +0.0071 | -0.0034 | -0.0052      | -0.0022 | +0.0090 |
| Normalized Accuracy ( $\Delta acc\_norm$ ) |              |         |         |              |         |         |              |         |         |
| ARC Easy                                   | -0.0080      | -0.0063 | +0.0084 | -0.0097      | -0.0055 | -0.0005 | -0.0054      | -0.0068 | +0.0164 |
| ARC Challenge                              | +0.0009      | +0.0290 | +0.0298 | +0.0034      | +0.0375 | +0.0102 | -0.0239      | +0.0214 | +0.0222 |
| SWAG                                       | +0.0038      | +0.0091 | +0.0003 | +0.0018      | +0.0093 | +0.0032 | +0.0015      | +0.0081 | -0.0009 |
| PIQA                                       | +0.0016      | +0.0033 | +0.0310 | -0.0033      | +0.0196 | +0.0294 | -0.0060      | -0.0071 | +0.0261 |
| HellaSwag                                  | +0.0045      | +0.0057 | +0.0079 | +0.0112      | +0.0065 | +0.0087 | +0.0063      | -0.0019 | +0.0015 |
| OpenBookQA                                 | +0.0380      | +0.1500 | +0.1220 | +0.0320      | +0.1440 | +0.1160 | +0.0380      | +0.1480 | +0.1080 |
| Avg.                                       | +0.0068      | +0.0318 | +0.0332 | +0.0059      | +0.0352 | +0.0278 | +0.0018      | +0.0269 | +0.0289 |

| Task                                       | Gemma3 270M |         |         | Qwen3 0.6B |         |         | Llama3.2 1B |         |         |
|--------------------------------------------|-------------|---------|---------|------------|---------|---------|-------------|---------|---------|
|                                            | 10k         | 50k     | 100k    | 10k        | 50k     | 100k    | 10k         | 50k     | 100k    |
| Accuracy ( $\Delta acc$ )                  |             |         |         |            |         |         |             |         |         |
| ARC Easy                                   | +0.0030     | +0.0088 | -0.0029 | +0.0017    | +0.0051 | -0.0109 | +0.0198     | +0.0217 | +0.0379 |
| ARC Challenge                              | -0.0248     | -0.0120 | +0.0085 | -0.0205    | -0.0214 | -0.0094 | -0.0470     | -0.0166 | -0.0111 |
| SWAG                                       | +0.0039     | +0.0093 | +0.0054 | +0.0133    | +0.0094 | +0.0050 | +0.0102     | +0.0094 | +0.0062 |
| PIQA                                       | +0.0055     | +0.0081 | +0.0054 | -0.0032    | -0.0038 | +0.0087 | +0.0211     | +0.0187 | -0.0044 |
| HellaSwag                                  | +0.0055     | +0.0068 | +0.0094 | +0.0087    | +0.0109 | +0.0122 | +0.0067     | +0.0067 | +0.0113 |
| OpenBookQA                                 | -0.0320     | +0.0180 | -0.0140 | -0.0240    | +0.0160 | -0.0220 | -0.0380     | +0.0040 | +0.0220 |
| Avg.                                       | -0.0065     | +0.0065 | +0.0020 | -0.0040    | +0.0027 | -0.0027 | -0.0045     | +0.0073 | +0.0103 |
| Normalized Accuracy ( $\Delta acc\_norm$ ) |             |         |         |            |         |         |             |         |         |
| ARC Easy                                   | -0.0038     | +0.0097 | +0.0173 | -0.0147    | +0.0059 | +0.0152 | +0.0047     | +0.0161 | +0.0240 |
| ARC Challenge                              | -0.0094     | +0.0324 | +0.0128 | +0.0111    | +0.0264 | +0.0145 | -0.0097     | +0.0383 | +0.0009 |
| SWAG                                       | +0.0003     | +0.0098 | +0.0026 | +0.0153    | +0.0045 | -0.0023 | +0.0084     | +0.0093 | +0.0015 |
| PIQA                                       | +0.0055     | +0.0103 | +0.0299 | +0.0141    | +0.0114 | +0.0380 | +0.0212     | +0.0203 | +0.0430 |
| HellaSwag                                  | +0.0053     | -0.0040 | +0.0041 | +0.0116    | +0.0045 | +0.0051 | -0.0028     | -0.0082 | -0.0035 |
| OpenBookQA                                 | +0.0280     | +0.1340 | +0.1260 | +0.0420    | +0.1640 | +0.1100 | +0.0480     | +0.1120 | +0.0700 |
| Avg.                                       | +0.0043     | +0.0320 | +0.0321 | +0.0132    | +0.0361 | +0.0301 | +0.0116     | +0.0313 | +0.0226 |

Table 1: English performance differences between word-level and SentencePiece Unigram across model configurations and target vocabulary sizes. Values are word-level minus Unigram; positive values favor word-level. Averages are over six tasks, and vocabulary sizes include reserved tokens. `acc_norm` uses candidate character length, not token count.

interact with sequence length, training dynamics, and benchmark-specific candidate tokenization. We therefore interpret downstream performance separately from the measured changes in tokenization efficiency.

## 6 Tokenization Efficiency Analysis

We analyze tokenization efficiency under the fixed 20B emitted-token budget in the multilingual Llama3.2 1B experiments. For each tokenizer, we report raw documents and language-specific segmented units covered, tokens per segmented unit, characters and UTF-8 bytes per emitted token, and the proportion of byte-fallback tokens. Segmented

units follow the language-specific procedures in Section 3; unit-level quantities are therefore intended for within-language tokenizer comparisons.

SentencePiece Unigram is more token-efficient than the word-level tokenizer in all 12 language-vocabulary settings (Table 3). It processes more raw documents and segmented units and represents more characters and UTF-8 bytes per emitted token under the same emitted-token budget.

Increasing the word-level vocabulary size reduces byte-fallback usage in all four languages and correspondingly reduces the number of emitted tokens per segmented unit, increasing raw-text coverage.

| Llama3.2 1B EN                             |         |         |         | Llama3.2 1B FR    |         |         |         |
|--------------------------------------------|---------|---------|---------|-------------------|---------|---------|---------|
| Benchmark                                  | 10k     | 50k     | 100k    | Benchmark         | 10k     | 50k     | 100k    |
| Accuracy ( $\Delta acc$ )                  |         |         |         |                   |         |         |         |
| ARC Easy                                   | +0.0198 | +0.0217 | +0.0379 | ARC Easy          | -0.0308 | +0.0111 | -0.0248 |
| ARC Challenge                              | -0.0470 | -0.0166 | -0.0111 | ARC Challenge     | -0.0333 | +0.0026 | -0.0196 |
| SWAG                                       | +0.0102 | +0.0094 | +0.0062 | PIQA              | -0.0300 | -0.0300 | +0.0500 |
| PIQA                                       | +0.0211 | +0.0187 | -0.0044 | HellaSwag         | -0.0021 | +0.0022 | +0.0003 |
| HellaSwag                                  | +0.0067 | +0.0067 | +0.0113 | French Grammar    | +0.1933 | +0.1345 | +0.0504 |
| OpenBookQA                                 | -0.0380 | +0.0040 | +0.0220 | French Vocabulary | +0.1428 | +0.1429 | +0.1765 |
| Avg.                                       | -0.0045 | +0.0073 | +0.0103 | Avg.              | +0.0400 | +0.0439 | +0.0388 |
| Normalized Accuracy ( $\Delta acc\_norm$ ) |         |         |         |                   |         |         |         |
| ARC Easy                                   | +0.0047 | +0.0161 | +0.0240 | ARC Easy          | -0.0008 | +0.0085 | +0.0035 |
| ARC Challenge                              | -0.0097 | +0.0383 | +0.0009 | ARC Challenge     | -0.0051 | +0.0068 | +0.0025 |
| SWAG                                       | +0.0084 | +0.0093 | +0.0015 | PIQA              | -0.0400 | +0.0200 | +0.0600 |
| PIQA                                       | +0.0212 | +0.0203 | +0.0430 | HellaSwag         | +0.0066 | +0.0015 | +0.0015 |
| HellaSwag                                  | -0.0028 | -0.0082 | -0.0035 | French Grammar    | +0.2437 | +0.1513 | +0.0420 |
| OpenBookQA                                 | +0.0480 | +0.1120 | +0.0700 | French Vocabulary | +0.1933 | +0.1008 | +0.0588 |
| Avg.                                       | +0.0116 | +0.0313 | +0.0226 | Avg.              | +0.0663 | +0.0482 | +0.0281 |

| Llama3.2 1B ZH                             |         |         |         | Llama3.2 1B JA |         |         |         |
|--------------------------------------------|---------|---------|---------|----------------|---------|---------|---------|
| Benchmark                                  | 10k     | 50k     | 100k    | Benchmark      | 10k     | 50k     | 100k    |
| Accuracy ( $\Delta acc$ )                  |         |         |         |                |         |         |         |
| Belebele                                   | +0.0133 | -0.0078 | -0.0155 | Belebele       | +0.0033 | -0.0056 | +0.0345 |
| ARC                                        | -0.0188 | -0.0145 | +0.0017 | JARC           | +0.0100 | +0.0300 | +0.0100 |
| PIQA                                       | +0.0100 | +0.0800 | +0.0500 | PIQA           | +0.0800 | -0.0400 | -0.0300 |
| PAWS-X                                     | +0.0085 | +0.0070 | +0.0110 | PAWS-X         | -0.0310 | +0.0840 | +0.0755 |
| XCOPA                                      | +0.0220 | +0.0080 | +0.0000 | Japanese COPA  | -0.0300 | -0.0040 | +0.0060 |
| XWinograd                                  | +0.0040 | +0.0218 | +0.0318 | XWinograd      | +0.0251 | +0.0177 | -0.0208 |
| Avg.                                       | +0.0065 | +0.0158 | +0.0132 | Avg.           | +0.0096 | +0.0137 | +0.0125 |
| Normalized Accuracy ( $\Delta acc\_norm$ ) |         |         |         |                |         |         |         |
| Belebele                                   | -0.0055 | +0.0022 | +0.0212 | Belebele       | -0.0134 | -0.0178 | +0.0034 |
| ARC                                        | -0.0154 | -0.0085 | +0.0000 | JARC           | +0.0600 | -0.0100 | +0.0000 |
| PIQA                                       | +0.0200 | +0.0500 | +0.0100 | PIQA           | +0.1200 | -0.0600 | +0.0600 |
| PAWS-X                                     | +0.0000 | -0.0165 | +0.0005 | PAWS-X         | -0.1165 | +0.0390 | +0.1065 |
| XCOPA                                      | +0.0040 | -0.0280 | +0.0160 | Japanese COPA  | +0.0060 | +0.0580 | +0.0020 |
| XWinograd                                  | -0.0020 | +0.0218 | +0.0139 | XWinograd      | +0.0021 | +0.0073 | -0.0042 |
| Avg.                                       | +0.0002 | +0.0035 | +0.0103 | Avg.           | +0.0097 | +0.0028 | +0.0280 |

Table 2: Multilingual performance differences between word-level and SentencePiece Unigram tokenization using Llama3.2 1B. Values are word-level minus Unigram, averaged over six tasks per language. Vocabulary sizes include reserved tokens. `acc_norm` uses candidate character length, not token count.

The character- and byte-based measures show the same efficiency pattern. SentencePiece Unigram represents more characters and UTF-8 bytes per emitted token than the word-level tokenizer in all 12 settings. Unlike `tokens/unit`, these measures do not depend on the language-specific definition of a segmented unit, providing a complementary view of tokenization efficiency.

Absolute byte rates should not, however, be compared directly across languages: OOV units are represented by UTF-8 bytes, whose number depends on the underlying writing system as well as vocabulary coverage.

Greater token efficiency does not consistently correspond to higher downstream accuracy. Although

Unigram covers more documents in all 12 settings, the average word-level raw-accuracy difference is positive in 11 of the 12 settings (Table 2). Thus, under the evaluated fixed-token-budget setting, greater raw-text coverage alone does not determine downstream performance.

## 7 Conclusion

We revisited word-level tokenization for language model pretraining under open-vocabulary conditions. By combining language-appropriate word-like segmentation with byte fallback, we compare word-level tokenization with SentencePiece Unigram without the conventional limitation of mapping OOV units to a single unknown token.

| Metric            | Type    | English |       |       | French |       |       |
|-------------------|---------|---------|-------|-------|--------|-------|-------|
|                   |         | 10k     | 50k   | 100k  | 10k    | 50k   | 100k  |
| docs_covered (M)  | Word    | 7.43    | 9.42  | 10.18 | 12.63  | 15.70 | 16.42 |
|                   | Unigram | 16.84   | 19.89 | 20.61 | 25.23  | 29.66 | 30.52 |
| units_covered (B) | Word    | 11.34   | 14.38 | 15.51 | 14.42  | 17.92 | 18.75 |
|                   | Unigram | 25.53   | 30.10 | 31.16 | 28.78  | 33.83 | 34.78 |
| tokens/unit       | Word    | 1.76    | 1.39  | 1.29  | 1.39   | 1.12  | 1.07  |
|                   | Unigram | 0.78    | 0.66  | 0.64  | 0.69   | 0.59  | 0.58  |
| chars/token       | Word    | 1.75    | 2.21  | 2.39  | 2.00   | 2.49  | 2.60  |
|                   | Unigram | 3.93    | 4.63  | 4.80  | 4.00   | 4.70  | 4.83  |
| bytes/token       | Word    | 1.76    | 2.23  | 2.40  | 2.06   | 2.56  | 2.68  |
|                   | Unigram | 3.95    | 4.66  | 4.82  | 4.11   | 4.83  | 4.97  |
| byte_rate (%)     | Word    | 49.45   | 31.78 | 25.29 | 31.81  | 11.84 | 7.13  |
|                   | Unigram | 1.78    | 2.10  | 2.18  | 0.05   | 0.06  | 0.06  |

| Metric            | Type    | Chinese |       |       | Japanese |       |       |
|-------------------|---------|---------|-------|-------|----------|-------|-------|
|                   |         | 10k     | 50k   | 100k  | 10k      | 50k   | 100k  |
| docs_covered (M)  | Word    | 9.72    | 12.80 | 14.61 | 14.76    | 19.26 | 20.38 |
|                   | Unigram | 14.41   | 20.44 | 21.97 | 21.67    | 29.75 | 32.23 |
| units_covered (B) | Word    | 8.36    | 11.03 | 12.59 | 13.50    | 17.62 | 18.65 |
|                   | Unigram | 12.43   | 17.64 | 18.97 | 19.82    | 27.28 | 29.75 |
| tokens/unit       | Word    | 2.39    | 1.81  | 1.59  | 1.48     | 1.14  | 1.07  |
|                   | Unigram | 1.61    | 1.13  | 1.05  | 1.01     | 0.73  | 0.67  |
| chars/token       | Word    | 0.71    | 0.93  | 1.06  | 1.11     | 1.45  | 1.54  |
|                   | Unigram | 1.05    | 1.49  | 1.60  | 1.63     | 2.25  | 2.45  |
| bytes/token       | Word    | 1.96    | 2.58  | 2.95  | 3.13     | 4.08  | 4.32  |
|                   | Unigram | 2.91    | 4.13  | 4.44  | 4.59     | 6.32  | 6.90  |
| byte_rate (%)     | Word    | 68.79   | 52.73 | 43.49 | 37.09    | 13.38 | 7.51  |
|                   | Unigram | 2.13    | 3.02  | 3.27  | 3.25     | 4.49  | 4.91  |

Table 3: Tokenization characteristics under a fixed 20B emitted-token budget for multilingual Llama3.2 1B. docs\_covered and units\_covered measure raw-text exposure; tokens/unit, chars/token, and bytes/token measure tokenization efficiency; and byte\_rate is the percentage of byte-fallback tokens. tokens/unit can be below one for SentencePiece because its tokens may cross word-like unit boundaries. Each condition uses its own corpus prefix up to the token budget. Vocabulary sizes denote target settings.

In the English experiments, raw-accuracy differences vary across model configurations and vocabulary sizes, with neither tokenizer showing a consistent overall advantage. Character-length-normalized accuracy favors word-level tokenization on average across all 18 architecture–vocabulary settings. Its denominator is based on candidate character length rather than tokenizer-specific token count, so we interpret it as a complementary downstream metric rather than evidence of a universal advantage in language modeling quality.

The multilingual experiments similarly show that word-level tokenization remains competitive in French, Chinese, and Japanese with language-appropriate word-like segmentation. Positive average raw-accuracy differences are observed at all three vocabulary sizes in these languages, including Chinese and Japanese, where whitespace alone does not determine word boundaries.

The efficiency analysis shows a clear trade-off. SentencePiece Unigram processes more raw text

under the same emitted-token budget in all 12 multilingual language–vocabulary settings, while increasing the word-level vocabulary consistently reduces byte-fallback usage and increases raw-text coverage. However, greater token efficiency does not consistently correspond to higher downstream accuracy.

Overall, our results do not establish a universal advantage for either word-level or subword tokenization. Rather, word-level tokenization can achieve competitive downstream performance when open-vocabulary coverage and language-appropriate word-like segmentation are provided, despite being less token-efficient in the evaluated settings. Additional SentencePiece BPE comparisons in Appendix C show a broadly similar pattern, with some differences across languages and vocabulary sizes.

These findings motivate further investigation of word-like lexical units as a viable alternative tokenization strategy for modern language models.

## Limitations

Our analysis relies on externally supplied language-specific segmentation procedures, which should not be interpreted as providing a unique linguistic definition of a word. English and French use rule-based segmentation, while Chinese and Japanese use jieba and MeCab, respectively. The resulting units depend on the segmentation rules, software, and dictionary choices used for each language. Alternative segmentation procedures could therefore produce different vocabularies and downstream results.

The comparison also jointly varies how segmentation boundaries are obtained. The word-level tokenizer relies on externally supplied language-specific segmentation, whereas SentencePiece learns its segmentation from the tokenizer-training corpus. Our experiments therefore compare these tokenizer designs as complete systems rather than isolating the effect of lexical unit size independently of the segmentation procedure.

The fixed emitted-token budget does not control the amount of raw text observed by each model. As shown in Section 6, SentencePiece Unigram processes more documents and segmented units than the corresponding word-level tokenizer under the same 20B emitted-token budget in all multilingual settings. Our experiments therefore do not isolate the effect of tokenization strategy from differences in raw-text exposure. Experiments with matched raw-text or document budgets would provide a complementary comparison.

Byte-fallback statistics also require care when compared across languages. Because an OOV segmented unit is represented as a sequence of UTF-8 bytes, the number of emitted byte tokens depends on both vocabulary coverage and the byte representation of the underlying text. Absolute byte rates should therefore not be interpreted as directly comparable measures of segmentation quality across languages.

Normalized accuracy divides candidate log-likelihood by the character length of the candidate string, not by its number of tokens. Although this denominator is shared across tokenizers for a given candidate string, the underlying token sequences and model likelihoods can differ. We therefore treat raw accuracy as the primary downstream metric and character-length-normalized accuracy as a complementary measure. The normalized accuracy results should not be interpreted as establishing a universal

advantage in language modeling quality.

Finally, the multilingual experiments use only the Llama3.2 1B architecture and a single pretraining run for each tokenizer condition. Although the English experiments cover six model configurations from four architecture families, the study remains limited in model scale, training budgets, datasets, languages, and segmentation procedures. Additional random seeds, larger models, longer training budgets, more languages, and alternative language-specific segmentation procedures would help assess the generality of the observed patterns.

## Ethical Considerations

This study does not involve human-subject experiments or author-collected personal information. However, the tokenizers and models are trained on large-scale text corpora that may contain biased, offensive, toxic, or otherwise harmful content. Because the word-level vocabulary is constructed from frequent segmented units, such content may also appear as explicit vocabulary entries.

Tokenizer behavior and efficiency can vary across languages and segmentation procedures. As observed in our experiments, vocabulary coverage and reliance on byte fallback differ across the evaluated languages, although absolute byte-fallback rates should not be directly interpreted as measures of linguistic quality or fairness. Such differences may lead to unequal tokenization efficiency or effective raw-text exposure across languages under a fixed token budget. Our results should therefore not be interpreted as establishing that any tokenization strategy is uniformly appropriate across languages or user populations.

Applications of such tokenizers and models should consider corpus filtering, vocabulary auditing, downstream safety evaluation, and evaluation across relevant languages and user groups.

## Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP26H02511, by the “R&D Hub Aimed at Ensuring Transparency and Reliability of Generative AI Models” project of the Ministry of Education, Culture, Sports, Science and Technology, and by JST Moonshot R&D Grant Number JPMJMS2011-35 (fundamental research). A part of this work was conducted using the Grand Chariot 2 supercomputer at the Information Initiative Center, Hokkaido University.

## References

- Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveiling, 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, and 2 others. 2024. [Tokenizer choice for LLM training: Negligible or crucial?](#) In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico. Association for Computational Linguistics.
- Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabza. 2024. [The belebele benchmark: a parallel reading comprehension dataset in 122 language variants](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 749–775, Bangkok, Thailand. Association for Computational Linguistics.
- Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, and 3 others. 2025. [SmolLM2: When Smol Goes Big—Data-Centric Training of a Small Language Model](#). *arXiv preprint arXiv:2502.02737*.
- Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. 2024. SmolLM-Corpus. <https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus>. Dataset.
- Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julien Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, and 11 others. 2024. [Lessons from the trenches on reproducible evaluation of language models](#). *arXiv preprint arXiv:2405.14782*.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. [PIQA: Reasoning about Physical Commonsense in Natural Language](#). In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 7432–7439. AAAI Press.
- Maharaj Brahma, N J Karthika, Rajat Verma, Nagasai Saketh Naidu, Rohit Saluja, Maunendra Sankar Desarkar, and Ganesh Ramakrishnan. 2026. [Multilingual tokenization through the lens of Indian languages: Challenges and insights](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 32614–32632, San Diego, California, United States. Association for Computational Linguistics.
- Yekun Chai, Yewei Fang, Qiwei Peng, and Xuhong Li. 2024. [Tokenization Falling Short: On Subword Robustness in Large Language Models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1582–1599, Miami, Florida, USA. Association for Computational Linguistics.
- Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. [CANINE: Pre-training an efficient tokenization-free encoder for language representation](#). *Transactions of the Association for Computational Linguistics*, 10:73–91.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. [Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge](#). *arXiv preprint arXiv:1803.05457*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). In *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. Association for Computational Linguistics.
- Manuel Faysse, Patrick Fernandes, Nuno Miguel Guerreiro, António Loison, Duarte M. Alves, Caio Corro, Nicolas Boizard, João Alves, Ricardo Rei, Pedro Henrique Martins, Antoni Bigata Casademunt, François Yvon, André F. T. Martins, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. [CroissantLLM: A truly bilingual french-english language model](#). *Trans. Mach. Learn. Res.*, 2025.
- Darius Feher, Ivan Vulić, and Benjamin Minixhofer. 2025. [Retrofitting Large Language Models with Dynamic Tokenization](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29866–29883, Vienna, Austria. Association for Computational Linguistics.
- fxsjy. n.d. jieba: Chinese text segmentation. <https://github.com/fxsjy/jieba>. Software.
- Matthias Gallé. 2019. [Investigating the effectiveness of BPE: The power of shorter sequences](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1375–1381, Hong Kong, China. Association for Computational Linguistics.
- Gemma Team. 2025. [Gemma 3 technical report](#). *CoRR*, abs/2503.19786.
- Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. [On Using Very Large Target Vocabulary for Neural Machine Translation](#).

- In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.
- Taku Kudo. 2018. **Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates**. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.
- Taku Kudo and John Richardson. 2018. **SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing**. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.
- Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. **Applying conditional random fields to Japanese morphological analysis**. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.
- Tomasz Limisiewicz, Jiří Balhar, and David Mareček. 2023. **Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages**. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5661–5681, Toronto, Canada. Association for Computational Linguistics.
- Llama Team. 2024. **The Llama 3 herd of models**. *CoRR*, abs/2407.21783.
- Ilya Loshchilov and Frank Hutter. 2019. **Decoupled weight decay regularization**. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.
- Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. **Effective Approaches to Attention-based Neural Machine Translation**. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.
- Paul McCann. 2020. **fugashi, a tool for tokenizing Japanese in Python**. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 44–51, Online. Association for Computational Linguistics.
- Meta AI. 2024. **Llama 3.2 Model Card**. Official model card.
- 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. 2021. **Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP**. *arXiv preprint arXiv:2112.10508*.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. **Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering**. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.
- nlp-titech. n.d. Japanese COPA dataset. <https://github.com/nlp-titech/copa-japanese>. Dataset.
- 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. 2025. **Byte latent transformer: Patches scale better than tokens**. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258, Vienna, Austria. Association for Computational Linguistics.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben Alal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2024. **The FineWeb datasets: Decanting the web for the finest text data at scale**. In *Advances in Neural Information Processing Systems*, volume 37, pages 30811–30849.
- Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Karagan, Colin Raffel, Martin Jaggi, Leandro von Werra, and Thomas Wolf. 2025. **FineWeb2: One pipeline to scale them all—adapting pre-training data processing to every language**. *arXiv preprint arXiv:2506.20920*.
- Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020. **XCOPA: A multilingual dataset for causal common-sense reasoning**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2362–2376, Online. Association for Computational Linguistics.
- Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. **BPE-Dropout: Simple and Effective Subword Regularization**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.
- Qwen Team. 2025. **Qwen3 technical report**. *CoRR*, abs/2505.09388.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. **Exploring the limits of transfer learning with a unified text-to-text transformer**. *Journal of Machine Learning Research*, 21(140):1–67.

- Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. [Choice of plausible alternatives: An evaluation of commonsense causal reasoning](#). In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*.
- David Samuel and Lilja Øvrelid. 2023. [Tokenization with factorized subword encoding](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14143–14161, Toronto, Canada. Association for Computational Linguistics.
- Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. [Tokenization is more than compression](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 678–702, Miami, Florida, USA. Association for Computational Linguistics.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Neural Machine Translation of Rare Words with Subword Units](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. 2021. [Fast WordPiece tokenization](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2089–2103, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Sho Takase, Ryokan Ri, Shun Kiyono, and Takuya Kato. 2025. [Large vocabulary size improves large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1015–1026, Vienna, Austria. Association for Computational Linguistics.
- Yuichi Tateno. 2025. [FineWeb2 Edu Japanese](https://huggingface.co/datasets/hotchpotch/fineweb-2-edu-japanese/). <https://huggingface.co/datasets/hotchpotch/fineweb-2-edu-japanese/>. Dataset.
- Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. [Charformer: Fast character transformers via gradient-based subword tokenization](#). In *International Conference on Learning Representations*.
- Avijit Thawani, Saurabh Ghanekar, Xiaoyuan Zhu, and Jay Pujara. 2023. [Learn Your Tokens: Word-Pooled Tokenization for Language Modeling](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9883–9893, Singapore. Association for Computational Linguistics.
- Alexey Tikhonov and Max Ryabinin. 2021. [It’s All in the Heads: Using Attention Heads as a Baseline for Cross-Lingual Transfer in Commonsense Reasoning](#). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3534–3546, Online. Association for Computational Linguistics.
- Cagri Toraman, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozcelik. 2023. Impact of Tokenization on Language Models: An Analysis for Turkish. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(4):1–21.
- weblab-GENIAC. n.d. JARC: Japanese translation of the AI2 reasoning challenge. <https://huggingface.co/datasets/weblab-GENIAC/jarc>. Dataset.
- Anna Wegmann, Dong Nguyen, and David Jurgens. 2025. [Tokenization is sensitive to language variation](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10958–10983, Vienna, Austria. Association for Computational Linguistics.
- Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. 2021. [Vocabulary learning via optimal transport for neural machine translation](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7361–7373, Online. Association for Computational Linguistics.
- Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. [ByT5: Towards a token-free future with pre-trained byte-to-byte models](#). *Transactions of the Association for Computational Linguistics*, 10:291–306.
- Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. [PAWS-X: A cross-lingual adversarial dataset for paraphrase identification](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3687–3692, Hong Kong, China. Association for Computational Linguistics.
- Lili Yu, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. MEGABYTE: Predicting million-byte sequences with multiscale transformers. In *Advances in Neural Information Processing Systems*, volume 36.
- Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. [SWAG: A large-scale adversarial dataset for grounded commonsense inference](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a Machine Really Finish Your Sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

## A Tokenizer Implementation Details

This appendix provides additional implementation details for the language-aware word-level tokenizers described in Section 3. For each language, the tokenizer applies a language-specific procedure to obtain word-like units. The resulting units are then independently mapped either to lexical vocabulary entries or to UTF-8 byte-fallback tokens. The procedures described below specify how these units are obtained and represented.

### A.1 English

For English, we use deterministic rule-based segmentation. Consecutive ASCII alphabetic characters form a single unit, digit sequences are split into individual digits, and other non-whitespace characters are treated as individual units. Whitespace is handled separately: each whitespace character is represented by the dedicated whitespace token, `_`. Thus, consecutive whitespace characters produce multiple whitespace tokens, although different whitespace characters are not distinguished by their token representations.

For non-whitespace units, vocabulary lookup is performed using the lowercased form. If the lowercased unit is present in the vocabulary, it is emitted as a single lexical token. Otherwise, the unit is represented by the UTF-8 bytes of its form before this lookup-time lowercasing. This procedure provides deterministic word-like boundaries without relying on a learned segmentation model.

### A.2 French

For French, we apply language-specific normalization before segmentation. The input is first normalized to Unicode NFC, selected apostrophe variants are mapped to the ASCII apostrophe, and the text is lowercased. Consecutive whitespace characters are then collapsed into a single space, and leading and trailing whitespace is removed.

The normalized text is segmented using a rule-based procedure that recognizes Unicode alphabetic sequences, including accented letters and internal apostrophes. Digit sequences are split into individual digits, and other non-whitespace characters are treated individually. Whitespace is represented by the dedicated whitespace token, `_`, after the preceding normalization.

The segmenter also separates common apostrophe-based clitics from the following unit. For example, `l'amour` is divided into `l'` and

`amour`. A small set of lexical exceptions, including `aujourd'hui` and `quelqu'un`, is retained as a single unit when encountered as a whole unit.

Vocabulary lookup is performed on the resulting normalized units. An OOV unit is represented by the UTF-8 bytes of its normalized form. Consequently, byte fallback retains the normalized unit rather than recovering capitalization or apostrophe variants removed during preprocessing.

### A.3 Chinese

For Chinese, we first separate whitespace and non-whitespace spans. Each whitespace character is represented by the dedicated whitespace token, `_`. Each non-whitespace span is then segmented using `jieba` (fxsjy, n.d.) with `cut_all=False`.

The resulting units are independently looked up in the vocabulary. An in-vocabulary unit is represented by a single lexical token, whereas an OOV unit is represented by its UTF-8 byte sequence. The tokenizer does not apply additional lowercasing, Unicode normalization, or digit-specific splitting internally. Instead, the boundaries of non-whitespace units follow the output of `jieba`.

### A.4 Japanese

For Japanese, we likewise separate whitespace and non-whitespace spans before morphological segmentation. Each whitespace character is represented by the dedicated whitespace token, `_`. Each non-whitespace span is analyzed using `MeCab` (Kudo et al., 2004) through `fugashi` (McCann, 2020).

For each resulting morphological unit, we use its surface form for vocabulary lookup rather than its lemma or other dictionary features. An in-vocabulary surface form is represented by a single lexical token, whereas an OOV surface form is represented by its UTF-8 byte sequence. The tokenizer does not apply additional lowercasing, Unicode normalization, or digit-specific splitting internally.

The `MeCab` dictionary location can be specified through the `MECAB_DICDIR` environment variable. The implementation provides a default path to a `UniDic-lite` dictionary when this variable is not set.

### A.5 Vocabulary Construction

For all languages, we apply the corresponding segmentation procedure to the tokenizer-training corpus and rank the resulting units by frequency. The word-level vocabulary includes special tokens, a

dedicated whitespace token, 256 byte-fallback tokens, and frequent segmented units selected according to the vocabulary-size setting. The same language-specific segmentation and vocabulary-construction procedure is used across the three nominal target vocabulary-size settings of 10k, 50k, and 100k.

The tokenizer-training data are distinct from the model-pretraining streams in their purpose and may also differ in dataset provenance. The available language-specific corpus specifications are as follows.

For English, the 10k, 50k, and 100k word-level vocabularies use the same deterministic segmentation and lookup procedure described in Appendix A.1. The released tokenizer assets preserve the vocabularies and tokenization implementation used in the experiments. The corresponding SentencePiece Unigram tokenizers use the same nominal vocabulary-size settings and support byte fallback.

For French, we construct the tokenizer-training corpus from five million non-empty documents obtained from the `fra_Latn` configuration of the FineWeb2 training split. Documents are read through streaming with a shuffle buffer of 50,000 and a random seed of 42. Before vocabulary construction, we apply Unicode NFC normalization, apostrophe normalization, lowercasing, and whitespace normalization. The word-level tokenizer and SentencePiece Unigram are constructed using the resulting normalized corpus.

For Chinese, the tokenizer-training corpus consists of five million documents from `opencsg/chinese-fineweb-edu-v2`. This differs from the `cmn_Hani` subset of FineWeb2 used for model pretraining. The tokenizer-training text is split into chunks of at most 4,000 Unicode characters.

For Japanese, we construct the tokenizer-training corpus from five million non-empty documents obtained from the training split of FineWeb2-Edu Japanese. The documents are read sequentially in streaming mode without explicit shuffling. After carriage-return normalization, null-character removal, and conversion of line breaks to spaces, each document is divided into chunks of at most 4,000 Unicode characters. The same chunked corpus is used for both MeCab-based word-level vocabulary construction and SentencePiece Unigram training.

At tokenization time, each segmented unit is considered independently. An in-vocabulary unit is represented by a single lexical token. An OOV unit

is instead encoded as a sequence of UTF-8 bytes, with each byte represented by its corresponding byte-fallback token. The byte sequence is formed from the unit produced by the applicable language-specific procedure, subject to the lookup-time lowercasing behavior described above for English.

This design separates the choice of word-like boundaries from the representation of units absent from the lexical vocabulary. Rather than replacing an OOV unit with a single unknown token, byte fallback provides a representation of that unit at the byte level. Here, “word-level” refers to the use of externally supplied word-like segmentation, not to a universal linguistic definition of a word.

## A.6 SentencePiece Baseline

The SentencePiece Unigram tokenizer used as the primary subword baseline is trained on the corresponding language-specific tokenizer-training corpus. Unlike the word-level tokenizer, which uses externally supplied segmentation boundaries, SentencePiece Unigram learns its segmentation statistically from the training corpus. As a result, a unit identified by the language-specific word-level procedure may be represented by one or more SentencePiece tokens.

Byte fallback is enabled for SentencePiece Unigram as well as for the word-level tokenizer. The comparison therefore examines externally specified word-like segmentation and learned subword segmentation under open-vocabulary tokenization conditions.

## B Pretraining Settings

Table 4 summarizes the pretraining configurations used in our experiments. Within each paired tokenizer comparison, the listed settings are held fixed while the tokenizer is varied.

All models are initialized from scratch, and model-family names refer only to architecture configurations rather than pretrained weights. Training budgets are defined in emitted tokenizer tokens and therefore do not imply equal raw-text exposure across tokenizers (Section 6).

All listed configurations use AdamW (Loshchilov and Hutter, 2019) with a cosine learning-rate schedule, a warmup ratio of 0.01, weight decay of 0.1, and a maximum gradient norm of 1.0. The training environment uses the PyTorch fused implementation of AdamW. The learning rate and batch size vary only where speci-

| Hyperparameter         | SmolLM2 135M       | SmolLM2 360M       | SmolLM2 1.7B       |
|------------------------|--------------------|--------------------|--------------------|
| Target training tokens | 2.7B               | 7.2B               | 34B                |
| Language               | English            | English            | English            |
| Sequence length        | 1024               | 1024               | 1024               |
| Batch size             | 8                  | 8                  | 4                  |
| Dataset                | FineWeb-Edu-Dedup  | FineWeb-Edu-Dedup  | FineWeb-Edu-Dedup  |
| Optimizer              | AdamW              | AdamW              | AdamW              |
| Learning rate          | $3 \times 10^{-5}$ | $3 \times 10^{-5}$ | $2 \times 10^{-5}$ |
| Warmup ratio           | 0.01               | 0.01               | 0.01               |
| Weight decay           | 0.1                | 0.1                | 0.1                |
| Max grad norm          | 1.0                | 1.0                | 1.0                |
| LR scheduler           | cosine             | cosine             | cosine             |

| Hyperparameter         | Gemma3 270M        | Qwen3 0.6B         | Llama3.2 1B EN     |
|------------------------|--------------------|--------------------|--------------------|
| Target training tokens | 5.4B               | 12B                | 20B                |
| Language               | English            | English            | English            |
| Sequence length        | 1024               | 1024               | 1024               |
| Batch size             | 8                  | 8                  | 8                  |
| Dataset                | FineWeb-Edu-Dedup  | FineWeb-Edu-Dedup  | FineWeb-Edu-Dedup  |
| Optimizer              | AdamW              | AdamW              | AdamW              |
| Learning rate          | $3 \times 10^{-5}$ | $3 \times 10^{-5}$ | $3 \times 10^{-5}$ |
| Warmup ratio           | 0.01               | 0.01               | 0.01               |
| Weight decay           | 0.1                | 0.1                | 0.1                |
| Max grad norm          | 1.0                | 1.0                | 1.0                |
| LR scheduler           | cosine             | cosine             | cosine             |

| Hyperparameter         | Llama3.2 1B FR     | Llama3.2 1B ZH     | Llama3.2 1B JA        |
|------------------------|--------------------|--------------------|-----------------------|
| Target training tokens | 20B                | 20B                | 20B                   |
| Language               | French             | Chinese            | Japanese              |
| Sequence length        | 1024               | 1024               | 1024                  |
| Batch size             | 8                  | 8                  | 8                     |
| Dataset                | FineWeb2 (French)  | FineWeb2 (Chinese) | FineWeb2-Edu Japanese |
| Optimizer              | AdamW              | AdamW              | AdamW                 |
| Learning rate          | $3 \times 10^{-5}$ | $3 \times 10^{-5}$ | $3 \times 10^{-5}$    |
| Warmup ratio           | 0.01               | 0.01               | 0.01                  |
| Weight decay           | 0.1                | 0.1                | 0.1                   |
| Max grad norm          | 1.0                | 1.0                | 1.0                   |
| LR scheduler           | cosine             | cosine             | cosine                |

Table 4: Pretraining settings for the model configurations and language settings used in our experiments. The upper block reports the English SmolLM2 scaling experiments, the middle block reports the additional English architecture experiments, and the lower block reports the multilingual Llama3.2 1B experiments. All models are trained from scratch, and model names denote architecture configurations rather than pretrained checkpoints. The training budget is defined in terms of emitted tokenizer tokens, and the same target budget is used within each paired tokenizer comparison. Batch sizes denote the global batch sizes used in the corresponding configurations.

fied in Table 4; otherwise, the listed optimization settings are shared across configurations.

Target vocabulary sizes are also matched between the word-level and SentencePiece conditions. Resulting token sequences and raw-text exposure may differ as consequences of the tokenizer, as quantified in Section 6.

## C BPE Comparison

As an additional comparison, we evaluate the word-level tokenizer against SentencePiece BPE using the same multilingual Llama3.2 1B setup as in the main experiments. The experiments use the

same languages, model architecture, target vocabulary sizes, pretraining corpora, preprocessing, and emitted-token budgets. Within each paired comparison, these experimental settings are held fixed while the tokenizer is varied.

Table 5 reports word-level minus BPE performance differences. Positive values favor word-level tokenization, whereas negative values favor SentencePiece BPE.

Raw accuracy favors word-level tokenization on average in 11 of the 12 language–vocabulary settings. English, French, and Japanese have positive average differences at all three vocabulary sizes.

| Llama3.2 1B EN                             |         |         |         | Llama3.2 1B FR    |         |         |         |
|--------------------------------------------|---------|---------|---------|-------------------|---------|---------|---------|
| Benchmark                                  | 10k     | 50k     | 100k    | Benchmark         | 10k     | 50k     | 100k    |
| Accuracy ( $\Delta acc$ )                  |         |         |         |                   |         |         |         |
| ARC Easy                                   | +0.0156 | +0.0061 | +0.0249 | ARC Easy          | -.0282  | +0.0000 | -.0119  |
| ARC Challenge                              | -.0274  | +0.0021 | -.0009  | ARC Challenge     | -.0291  | -.0051  | -.0179  |
| SWAG                                       | +0.0039 | -.0020  | +0.0014 | PIQA              | -.0400  | -.0500  | +0.0000 |
| PIQA                                       | +0.0108 | +0.0018 | -.0191  | HellaSwag         | +0.0014 | +0.0046 | +0.0014 |
| HellaSwag                                  | +0.0062 | -.0009  | +0.0059 | French Grammar    | +0.1765 | +0.0925 | +0.0588 |
| OpenBookQA                                 | -.0040  | +0.0020 | +0.0520 | French Vocabulary | +0.1344 | +0.1512 | +0.0840 |
| Avg.                                       | +0.0009 | +0.0015 | +0.0107 | Avg.              | +0.0358 | +0.0322 | +0.0191 |
| Normalized Accuracy ( $\Delta acc\_norm$ ) |         |         |         |                   |         |         |         |
| ARC Easy                                   | +0.0073 | +0.0111 | +0.0114 | ARC Easy          | -.0111  | +0.0017 | -.0256  |
| ARC Challenge                              | +0.0005 | +0.0169 | -.0059  | ARC Challenge     | -.0094  | -.0035  | -.0180  |
| SWAG                                       | +0.0031 | -.0083  | -.0083  | PIQA              | -.0100  | +0.0300 | +0.0500 |
| PIQA                                       | +0.0038 | +0.0029 | +0.0087 | HellaSwag         | +0.0029 | +0.0058 | +0.0001 |
| HellaSwag                                  | -.0241  | -.0252  | -.0173  | French Grammar    | +0.1681 | +0.1513 | +0.0756 |
| OpenBookQA                                 | +0.0040 | +0.0360 | +0.0500 | French Vocabulary | +0.1765 | +0.0840 | +0.0000 |
| Avg.                                       | -.0009  | +0.0056 | +0.0064 | Avg.              | +0.0528 | +0.0449 | +0.0137 |

| Llama3.2 1B ZH                             |         |         |         | Llama3.2 1B JA |         |         |         |
|--------------------------------------------|---------|---------|---------|----------------|---------|---------|---------|
| Benchmark                                  | 10k     | 50k     | 100k    | Benchmark      | 10k     | 50k     | 100k    |
| Accuracy ( $\Delta acc$ )                  |         |         |         |                |         |         |         |
| Belebele                                   | +0.0033 | -.0045  | +0.0178 | Belebele       | -.0045  | -.0045  | +0.0322 |
| ARC                                        | -.0119  | -.0094  | +0.0076 | JARC           | -.0600  | +0.0400 | +0.0400 |
| PIQA                                       | -.1100  | +0.0600 | +0.0300 | PIQA           | +0.0600 | -.0400  | -.0400  |
| PAWS-X                                     | -.0085  | -.0780  | +0.0370 | PAWS-X         | +0.0040 | +0.1050 | +0.0250 |
| XCOPA                                      | -.0120  | +0.0200 | -.0140  | Japanese COPA  | -.0140  | -.0180  | -.0040  |
| XWinograd                                  | +0.0159 | +0.0198 | -.0079  | XWinograd      | +0.0272 | +0.0010 | -.0146  |
| Avg.                                       | -.0205  | +0.0013 | +0.0118 | Avg.           | +0.0021 | +0.0139 | +0.0064 |
| Normalized Accuracy ( $\Delta acc\_norm$ ) |         |         |         |                |         |         |         |
| Belebele                                   | +0.0000 | +0.0111 | +0.0289 | Belebele       | -.0156  | -.0234  | +0.0000 |
| ARC                                        | -.0137  | +0.0009 | -.0017  | JARC           | -.0300  | +0.0700 | +0.0400 |
| PIQA                                       | -.1000  | +0.0600 | +0.0400 | PIQA           | +0.1100 | -.0700  | -.0500  |
| PAWS-X                                     | +0.0080 | -.0205  | +0.0015 | PAWS-X         | -.0235  | +0.1075 | +0.0030 |
| XCOPA                                      | -.0220  | -.0060  | -.0120  | Japanese COPA  | +0.0000 | +0.0340 | -.0020  |
| XWinograd                                  | -.0238  | +0.0040 | -.0119  | XWinograd      | +0.0052 | +0.0000 | +0.0000 |
| Avg.                                       | -.0253  | +0.0083 | +0.0075 | Avg.           | +0.0077 | +0.0197 | -.0015  |

Table 5: Word-level-BPE performance differences across four languages using Llama3.2 1B. Values are word-level minus BPE; positive values favor word-level tokenization. Averages are computed over the six evaluation tasks within each language. Vocabulary labels denote target vocabulary-size settings.  $acc\_norm$  normalizes candidate log-likelihood by candidate character length, not tokenizer-specific token count.

The exception is Chinese at 10k, where the average difference is  $-0.0205$ , favoring BPE. The magnitude of the positive differences varies across languages and vocabulary sizes, with particularly large averages in French.

The Chinese results also show a notable interaction with vocabulary size. The average raw-accuracy difference changes from  $-0.0205$  at 10k to  $+0.0013$  at 50k and  $+0.0118$  at 100k. Normalized accuracy follows the same broad pattern, changing from  $-0.0253$  at 10k to positive average differences at 50k and 100k. However, task-level differences remain mixed even when the average difference is positive.

Across all four languages, the average difference in normalized accuracy is positive in nine of the

12 settings. The exceptions are English at 10k, Chinese at 10k, and Japanese at 100k. As in the Unigram comparison, the BPE results vary across languages, vocabulary sizes, tasks, and evaluation metrics. Overall, these results show that word-level tokenization remains competitive beyond the Unigram baseline.

## D BPE Tokenization Efficiency

To complement the downstream results, we analyze SentencePiece BPE under the same fixed 20B emitted-token budget used in the multilingual Llama3.2 1B experiments. Table 6 reports the corresponding tokenization statistics alongside those of the word-level tokenizer. We use the same analysis procedure and language-specific definitions of

| Metric            | Type | English |       |       | French |       |       |
|-------------------|------|---------|-------|-------|--------|-------|-------|
|                   |      | 10k     | 50k   | 100k  | 10k    | 50k   | 100k  |
| docs_covered (M)  | Word | 7.43    | 9.42  | 10.18 | 12.63  | 15.70 | 16.42 |
|                   | BPE  | 17.54   | 20.20 | 20.68 | 25.00  | 29.59 | 30.53 |
| units_covered (B) | Word | 11.34   | 14.38 | 15.51 | 14.42  | 17.92 | 18.75 |
|                   | BPE  | 26.58   | 30.57 | 31.28 | 28.53  | 33.75 | 34.79 |
| tokens/unit       | Word | 1.76    | 1.39  | 1.29  | 1.39   | 1.12  | 1.07  |
|                   | BPE  | 0.75    | 0.65  | 0.64  | 0.70   | 0.59  | 0.57  |
| chars/token       | Word | 1.75    | 2.21  | 2.39  | 2.00   | 2.49  | 2.60  |
|                   | BPE  | 4.09    | 4.71  | 4.82  | 3.96   | 4.69  | 4.83  |
| bytes/token       | Word | 1.76    | 2.23  | 2.40  | 2.06   | 2.56  | 2.68  |
|                   | BPE  | 4.11    | 4.73  | 4.84  | 4.07   | 4.82  | 4.97  |
| byte_rate (%)     | Word | 49.45   | 31.78 | 25.29 | 31.81  | 11.84 | 7.13  |
|                   | BPE  | 5.17    | 5.73  | 5.86  | 0.05   | 0.06  | 0.06  |

| Metric            | Type | Chinese |       |       | Japanese |       |       |
|-------------------|------|---------|-------|-------|----------|-------|-------|
|                   |      | 10k     | 50k   | 100k  | 10k      | 50k   | 100k  |
| docs_covered (M)  | Word | 9.72    | 12.80 | 14.61 | 14.76    | 19.26 | 20.38 |
|                   | BPE  | 14.25   | 20.56 | 22.10 | 21.83    | 29.78 | 32.18 |
| units_covered (B) | Word | 8.36    | 11.03 | 12.59 | 13.50    | 17.62 | 18.65 |
|                   | BPE  | 12.30   | 17.74 | 19.08 | 19.97    | 27.32 | 29.70 |
| tokens/unit       | Word | 2.39    | 1.81  | 1.59  | 1.48     | 1.14  | 1.07  |
|                   | BPE  | 1.63    | 1.13  | 1.05  | 1.00     | 0.73  | 0.67  |
| chars/token       | Word | 0.71    | 0.93  | 1.06  | 1.11     | 1.45  | 1.54  |
|                   | BPE  | 1.04    | 1.50  | 1.61  | 1.65     | 2.25  | 2.45  |
| bytes/token       | Word | 1.96    | 2.58  | 2.95  | 3.13     | 4.08  | 4.32  |
|                   | BPE  | 2.88    | 4.15  | 4.46  | 4.63     | 6.33  | 6.88  |
| byte_rate (%)     | Word | 68.79   | 52.73 | 43.49 | 37.09    | 13.38 | 7.51  |
|                   | BPE  | 2.11    | 3.05  | 3.28  | 3.27     | 4.49  | 4.90  |

Table 6: Word-level and SentencePiece BPE tokenization characteristics under a fixed 20B emitted-token budget for multilingual Llama3.2 1B. docs\_covered and units\_covered measure raw-text exposure; tokens/unit, chars/token, and bytes/token measure efficiency; and byte\_rate denotes byte-fallback percentage. SentencePiece tokens may cross unit boundaries, allowing tokens/unit below one. Each condition uses its own corpus prefix. Definitions follow Table 3; vocabulary sizes denote target settings.

segmented units as in Section 6.

SentencePiece BPE is more token-efficient than the word-level tokenizer in all 12 language–vocabulary settings. It processes more raw documents and segmented units and represents more characters and UTF-8 bytes per emitted token. The word-level tokenizer also has a higher byte-fallback rate in every setting, although its byte-fallback rate decreases as vocabulary size increases. Thus, the greater raw-text coverage observed for Unigram in the main analysis is also present for BPE.

The effect of vocabulary size is also visible in the BPE comparison. For both tokenizers, increasing the target vocabulary size generally increases the amount of raw text processed under the fixed emitted-token budget. For example, BPE document coverage increases from 17.54M to 20.68M in English and from 21.83M to 32.18M in Japanese between 10k and 100k, while the corresponding word-level coverage increases from 7.43M to 10.18M and from 14.76M to 20.38M. Although the magni-

tude varies across languages, larger vocabularies generally represent the underlying text using fewer emitted tokens.

BPE and Unigram show broadly similar tokenization efficiency. At 100k, they process 20.68M and 20.61M documents in English and 32.18M and 32.23M in Japanese, respectively. Their character- and byte-based efficiency measures are also similar, although their relative ordering varies across languages and vocabulary sizes. Thus, neither sub-word method shows a uniform efficiency advantage.

Finally, greater token efficiency does not consistently correspond to higher downstream accuracy. Despite processing less raw text under the same emitted-token budget, word-level tokenization has positive raw-accuracy differences against BPE in 11 of 12 language–vocabulary settings (Table 5). Task-level results remain mixed across languages and vocabulary sizes, consistent with the distinction between tokenization efficiency and downstream performance observed with Unigram.