Benchmarking Byte-Pair Encoding Tokenizers on Different Languages with Bits per Byte

Soham Chowdhury[*1], **Warren Reagan Woolf**[*1]

[1]Independent Researcher
[*]Equal contribution

Correspondence: soham.chowdhury2020@gmail.com, warren.reagan.woolf@gmail.com

Abstract

Tokenization significantly affects the cross-lingual performance of language models, yet recent tokenizer variants such as SuperBPE and MorphBPE have not been systematically evaluated across typologically diverse languages. We conduct the first extrinsic cross-language comparison of BPE, SuperBPE, and MorphBPE tokenizers on English, Mandarin, and Hungarian, using bits per byte (BPB) normalized perplexity as our metric, with vocabulary sizes of 8K, 16K, and 32K. We find that SuperBPE matches BPE for English but underperforms by 0.01–0.06 BPB for Hungarian and Mandarin, suggesting that cross-whitespace merging is counterproductive for non-English languages. MorphBPE performs worse than BPE across all settings, with gaps of 0.02–0.04 BPB at the 32K vocabulary size. These results suggest that linguistic theory alone does not guarantee practical improvements in tokenizer design, and that standard BPE remains a surprisingly effective baseline across typologically diverse languages.[1]

1 Introduction

Tokenization is the process of converting text into a sequence of numbers in order to be able to efficiently feed them into machine learning models. The tokenizer algorithm can have a large effect on downstream performance of language models. Most modern tokenizers are essentially compression algorithms: The most popular tokenization algorithm, Byte-Pair Encoding (BPE), was originally designed to compress text (Gage, 1994). Intuitively, tokenization algorithms represent frequently occurring substrings as single tokens, while rare substrings are decomposed into sequences of smaller tokens.

Language model performance varies across languages, and this disparity stems in part from tokenizer design. The same passage of text, when translated into different languages and encoded with a tokenizer, might have drastically different token counts (Petrov et al., 2023). This problem is prominent in synthetic languages, because words in these languages frequently change forms, and the same root word appears in many surface forms. Due to this trait, many words might not even appear in the training corpora of these languages, and hence many tokenizers designed for analytic languages, especially English, have to split them into very small chunks. These splits are effective at compression, but are often morphologically incorrect.

There are two main ways to evaluate the performance of such tokenizers:

1. **Intrinsic Evaluation** measures the performance of tokenizers in isolation. Commonly used metrics include fertility, parity, and corpus token count.

2. **Extrinsic Evaluation** measures the performance of models trained using the given tokenizer on some downstream task. This is a reliable metric, since it directly measures the final performance.

Previously, researchers predominantly used intrinsic evaluation, since extrinsic evaluation is computationally expensive (Petrov et al., 2023). Unfortunately, recent work has shown that strong intrinsic performance is necessary but *not sufficient* for strong extrinsic performance (Ali et al., 2024).

In this work, we perform the first systematic extrinsic evaluation of the recent SuperBPE and MorphBPE tokenizers across English, Mandarin, and Hungarian using bits per byte (BPB) as a metric (Liu et al., 2025; Asgari et al., 2025). Pre-training performance seems to correlate well with performance on downstream tasks (Kaplan et al., 2020;

[1]Code available at github.com/warrenwoolf/multilingual-tokenizer-benchmarking.

Proceedings of the 1st Workshop on Multilinguality in the Era of Large Language Models (MeLLM 2026), pages 275–283

Du et al., 2025). Because a token in one language or tokenizer may encode more information than a token in another, raw perplexity comparisons are misleading. BPB normalizes by the total number of bytes, enabling fairer cross-tokenizer comparison.

2 Related Work

The relationship between tokenization and downstream model performance has attracted growing attention, yet the field lacks a consensus on how to evaluate tokenizers fairly across languages. Early work established fertility and compression ratio as the standard intrinsic metrics for tokenizer evaluation (Rust et al., 2021), but subsequent studies have challenged whether these metrics are sufficient. Schmidt et al. (2024) demonstrated that tokenization quality is not reducible to compression, and Ali et al. (2024) showed that compression-based metrics are unreliable for multilingual tokenizer selection. Goldman et al. (2024) found that better-compressing tokenizers correlate with improved downstream performance on generation tasks or for small models, though the correlation is weaker for classification tasks. More recently, Lotz et al. (2025) examined tokenizer evaluation across model scales, noting that the ranking of tokenizers by compression-based metrics does not always match their ranking by downstream task performance, motivating more careful cross-scale analysis.

The limitations of comparing perplexity across tokenizers have been independently recognized. Recently, Taylor and McCoy (2025) introduced the Weighted Perplexity Benchmark, demonstrating that tokenization differences can affect traditional perplexity measurements by up to 21.6% across 19 models, and proposing tokenizer-normalized evaluation frameworks to enable fair comparison. Their work reinforces the motivation for BPB normalization as used in our paper, while also highlighting that even BPB is not a perfect normalization.

Morphological alignment of tokenizers has been studied as a potential driver of the performance gap between English and morphologically rich languages. Arnett and Bergen (2025) investigated why language models perform worse on morphologically complex languages, introducing MorphScore to evaluate morphological alignment across 22 languages. They found that agglutinative languages tend to have less morphologically aligned tokenizations. Several studies argue that morphologically aligned tokenization is associated with improved performance across a variety of NLP tasks (Parra, 2024; Hofmann et al., 2021; Bauwens and Delobelle, 2024). However, the picture is not uniform: Arnett et al. (2024) found that different tokenization schemes lead to comparable performance for Spanish number agreement tasks. Moreover, Luitel et al. (2025) observed several counterintuitive patterns when investigating whether perplexity predicts fine-tuning performance for Nepali, a morphologically-complex language. Similarly, the original MorphBPE paper itself explicitly excluded extrinsic evaluation due to computational cost (Asgari et al., 2025), making our work a direct complement.

A parallel line of work has studied tokenization inequity across languages. Ahia et al. (2023) showed that languages incur drastically different costs when processed by commercial LLM tokenizers, with low-resource and morphologically complex languages requiring many more tokens to encode the same content. Petrov et al. (2023) also quantified this disparity across a broader set of languages and models, establishing the now-standard finding that tokenization length varies significantly across languages for equivalent semantic content.

3 Preliminaries

3.1 Tokenization Algorithms

Tokenizers are the core component that converts raw text into discrete units that models can process. Understanding how different tokenization approaches work is essential for understanding their performance across languages.

3.1.1 Byte-Pair Encoding (BPE)

Byte-Pair Encoding is an iterative compression algorithm that operates by repeatedly merging the most frequently occurring pair of tokens in a corpus (Sennrich et al., 2016). It starts with a vocabulary of individual bytes or characters and builds up larger subword units in the hope that frequently occurring byte sequences represent meaningful units. BPE assumes that token frequency is a good proxy for utility, which works well for analytic languages where common words naturally align with linguistic units. However, this assumption breaks down in morphologically complex languages where rare combinations of morphemes

are linguistically meaningful. Notably, most implementations forbid BPE from merging across whitespaces, which otherwise results in better compression but *worse* downstream performance (Liu et al., 2025).

3.1.2 SuperBPE

SuperBPE extends the standard BPE algorithm by adding a second phase of tokenizer training that merges tokens across whitespace boundaries. This results in significantly better performance on some downstream tasks. For example, Liu et al. (2025) found that SuperBPE yields a 20.3 percentage point improvement on the CommonsenseQA benchmark compared to a model that used BPE. Note that this benchmark is in English.

3.1.3 MorphBPE

MorphBPE is a morphology-aware variant of BPE that forbids merges across morpheme boundaries (Asgari et al., 2025). This restriction reduces the compression efficiency of the tokenizer, but lowers loss and perplexity, especially for synthetic languages.

3.2 Language Typology

Languages differ fundamentally in how they encode grammatical information and structure meaning. These differences have direct implications for how tokenizers perform.

Analytic languages like Mandarin, and, to a lesser degree, English, both express grammatical relationships primarily through word order and function words rather than inflectional morphology. Aside from the fact that most LLM research is focused on analytic languages, these languages are also simply very well-suited to tokenization. Words in these languages are typically short and undergo minimal inflectional change. Due to this, each word usually receives its own token in BPE, resulting in a relatively low token-to-word ratio close to 1.

Synthetic and agglutinative languages, such as Hungarian, encode grammatical information through morphological modification and character combination rather than word order. Words change form frequently, and the same root often appears in multiple surface forms. LLMs struggle when BPE greedily merges character combinations that do not form meaningful semantic units. This is where morphology-aware tokenizers such as MorphBPE are designed to excel.

3.3 Evaluation Metrics

Evaluating tokenizer quality requires metrics that capture both absolute compression performance and cross-tokenizer fairness.

3.3.1 Perplexity

Perplexity is the primary metric for language model performance on pre-training objectives (Shannon). It is defined as:

$$\text{PP} = e^{-\frac{1}{N}\sum_{i=1}^{N}\ln P(w_i)} \tag{1}$$

where N is the total number of tokens and $P(w_i)$ is the model's predicted probability of token i. Lower perplexity indicates better model performance. However, perplexity alone is unsuitable for cross-tokenizer comparison because the number of tokens varies across tokenizers for equivalent semantic content.

3.3.2 Bits Per Byte (BPB)

To address the cross-tokenizer comparison problem, we normalize perplexity by bits per byte, defined as:

$$\text{BPB} = \frac{-\sum_{i=1}^{N}\ln P(w_i)}{\ln(2)\,B_{\text{total}}} \tag{2}$$

where B_{total} is the total number of bytes in the corpus. This normalization accounts for the fact that different languages encode different amounts of information per byte. For instance, Mandarin Chinese packs substantially more semantic content per byte than English due to character density. By dividing by bytes rather than tokens, BPB provides a fairer cross-tokenizer comparison of how efficiently models compress information. Recent work has shown that even BPB normalization has limitations for perfectly fair comparison, but it remains the most practical metric for cheaply evaluating tokenizer performance across diverse languages (Arnett and Bergen, 2025). Notably, Liu et al. (2025) found that BPB rankings reverse when models are trained for long enough, but that this difference goes away when excluding the three most common words or when fixing both training and inference compute.

4 Experiments

4.1 Data and Languages

We used FineWeb and FineWeb2 for the English, Mandarin, and Hungarian languages Penedo et al. (2024, 2025). The training data consists of 100,000

documents per language for the training set and 5,000 documents per language for the test set. The training set contained approximately 300MB of text per language.

English, which is an analytic language with some synthetic properties, serves as our baseline, because most modern tokenizers are designed and trained on English-heavy corpora. Mandarin represents a fully analytic language, where morphological information is encoded through compounding and word order. Hungarian represents agglutinative and synthetic languages, where single roots combine with stacked suffixes to generate a combinatorial explosion of surface forms, most of which are not present in training data.

4.2 Tokenizers

We use three tokenization algorithms at vocabulary sizes of 8,000, 16,000, and 32,000 tokens. All tokenizers are trained on the same 100,000-document corpora for each language.

BPE: We use byte-level BPE rather than character-level BPE. This approach is conceptually similar to OpenAI's TikToken, as it operates on the bytes of text (OpenAI, 2023). Byte-level BPE handles out-of-vocabulary characters more gracefully than character-level approaches, particularly for non-Latin scripts.

SuperBPE: SuperBPE employs a two-stage training process. The first stage uses the full 300MB corpus (100,000 documents), identical to standard BPE. The second stage, which refines the vocabulary, uses only 200MB of data to slightly reduce computational overhead while preserving quality.

MorphBPE: MorphBPE is trained identically to BPE in terms of data and initialization, with the key difference being its morphology-aware merging strategy. We used MorphyNet for morpheme segmentation in English and Hungarian (Batsuren et al., 2021). We did not test MorphBPE with Mandarin, because most morphemes in Mandarin are words. We note a deviation from the original MorphBPE paper: our implementation treats whitespace and morpheme boundaries identically during training, whereas the original paper treats them separately. This is because we implemented MorphBPE by internally inserting spaces at all morpheme boundaries to force BPE to avoid merging across morpheme boundaries. This simplification reduces implementation complexity, but represents a slight departure from the original design. Also note an unnoticed bug that caused our MorphyNet-based morpheme segmenter to be case-sensitive, but which did not result in a significantly different BPB score.

English			
Tokenizer	**8K**	**16K**	**32K**
BPE	2.42	2.23	2.11
SuperBPE	2.43	2.24	2.10
MorphBPE	2.42	2.24	2.13
Hungarian			
Tokenizer	**8K**	**16K**	**32K**
BPE	2.41	2.24	2.10
SuperBPE	2.43	2.26	2.11
MorphBPE	2.44	2.27	2.14
Mandarin			
Tokenizer	**8K**	**16K**	**32K**
BPE	2.78	2.58	2.43
SuperBPE	2.80	2.62	2.49
MorphBPE	—	—	—

Table 1: BPB for English, Hungarian, and Mandarin. Lower is better.

4.3 Evaluation Metrics

We use BPB as an evaluation metric as defined in section 3.3.

5 Results

5.1 Overall Performance

We evaluate BPB scores (lower is better) for each tokenizer-vocabulary-language combination. Table 1 presents the primary results.

The scores for all tokenizers within a language and vocabulary size are very similar, but the differences may hold predictive value. Several patterns emerge. First, SuperBPE is approximately as good as BPE for English, but is consistently slightly worse for Hungarian and Mandarin. This suggests that cross-word merging is counterproductive for Hungarian, possibly because it is a synthetic language. Second, MorphBPE performs slightly worse than the alternatives in both tested languages. Notably, this is in spite of the fact that LMs using MorphBPE yielded a *lower* loss during training, as shown by Figure 1. This lower loss is consistent with the findings of the original paper (Asgari et al., 2025). The higher BPB suggests ei-

Language	Tokens
English: "The farmers and the workers are replanting the crops"	
BPE	The | farmers | and | the | workers | are | repl | ant | ing | the | crops
SuperBPE	The | farmers | and the | workers | are | repl | ant | ing the | crops
MorphBPE	The | farmer | s | and | the | worker | s | are | re | plant | ing | the | crop | s
Hungarian: "A gazdák és a munkások újraültetik a terményeket" *(The farmers and the workers are replanting the crops)*	
BPE	A | gazd | ák | és | a | munk | ások | újra | ült | etik | a | ter | mény | eket
SuperBPE	A | gazd | ák | és a | munk | ások | újra | ült | etik | a | ter | mény | eket
MorphBPE	A | gazda | k | és | a | munkás | ok | újra | ültet | ik | a | ter | mény | ek | et
Mandarin: "农民和工人们重新种植粮食" *("Nóngmín hé gōngrén men chóngxīn zhòngzhí liángshí", "Farmers and workers replant grain")*	
BPE	农民 | 和 | 工 | 人们 | 重新 | 种植 | 粮食
SuperBPE	*(same as BPE —Chinese corpus whitespace does not mark word boundaries)*
MorphBPE	*(not applicable)*

Table 2: Tokenization examples at 16K vocabulary. MorphBPE respects morpheme boundaries: *replanting* → re|plant|ing, *farmers* → farm|er|s. SuperBPE merges high-frequency cross-word pairs: English and the, ing the; Hungarian és a ("and the"). For Mandarin, SuperBPE produces identical output to BPE because whitespace in the Chinese web corpus marks HTML artefacts rather than linguistic word boundaries, leaving no useful structure for cross-word merges.

ther that constraining merges by morpheme boundaries harms pretraining efficiency or that our implementation choices treating morpheme boundaries identically to whitespace and relying on a case-sensitive segmenter introduced systematic disadvantages. However, we reran one MorphBPE training run with a case-insensitive segmenter, which yielded a BPB that was identical to the case-sensitive tokenizer to three decimal places.

5.2 Tokenization Examples

Table 2 shows representative tokenizations highlighting morphological splitting differences.

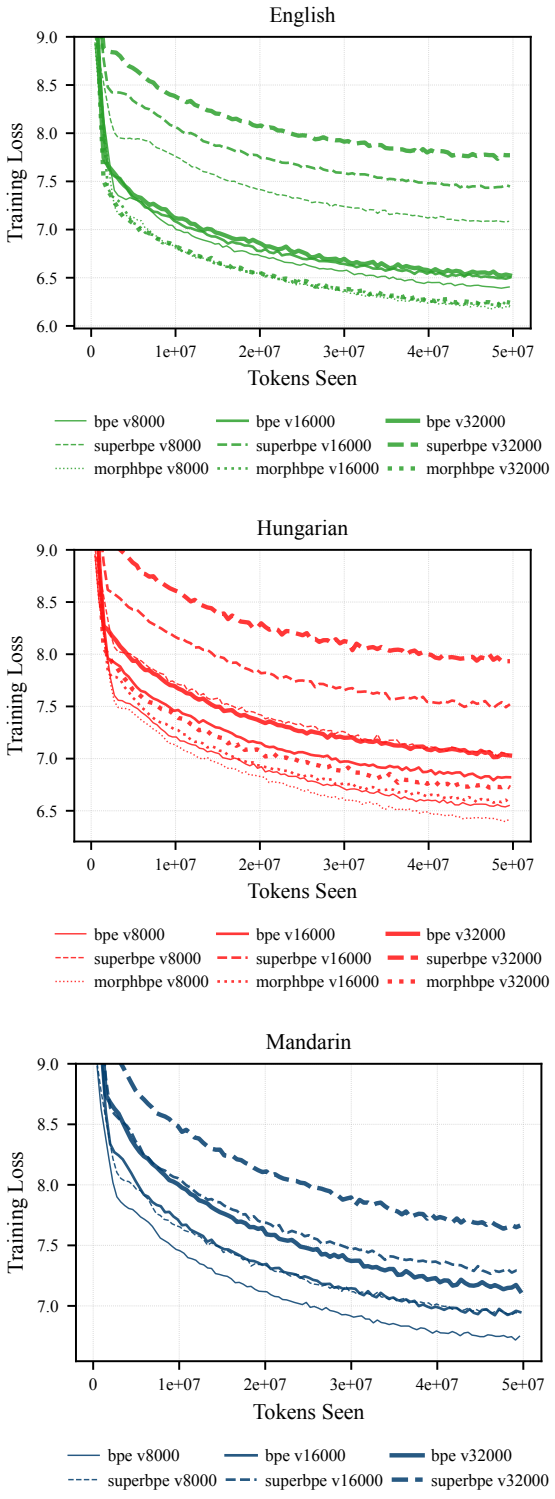

Figure 1: Training loss (lower is better) versus tokens seen across languages, tokenization algorithms, and vocabulary sizes. Note how MorphBPE gets a low loss and how SuperBPE gets a high loss.

5.3 Model Quality

Table 3 shows an example of a trained model's output. The output is incoherent because these

Prompt	The capital of France is
Completion	to the of, the of, the of, …

Table 3: Greedy completion from the BPE 32K English model (50M training tokens)

models are undertrained. Undertraining theoretically affects all tokenizers equally, so the BPB comparisons across them remain valid. Liu et al. (2025) shows that, when the 3 most common tokens are not excluded, SuperBPE BPB rankings invert when the models are trained for long enough, but our models are trained for 1.2×10^{16} FLOPs or less, which is before this switchover point is reached.

5.4 Training Stability

We ran the English training runs twice and found that BPB was identical to two decimal places across all 9 runs.

5.5 Summary

Our results reveal two primary findings:

1. **SuperBPE underperforms for Hungarian and Mandarin:** For English, SuperBPE is within 0.01 bits per byte of BPE. However, for Hungarian, SuperBPE has 0.01-0.02 more bits per byte than BPE, and for Mandarin, SuperBPE has 0.02-0.06 more bits per byte. This implies that, insofar as BPB predicts downstream performance, SuperBPE should only be used for analytic languages that separate words with whitespace, like English.

2. **MorphBPE underperforms in English and Hungarian:** While MorphBPE is only slightly worse than alternatives for low vocabulary sizes, at 32K it incurs an additional 0.02-0.04 more BPB for Hungarian and English. This implies that, insofar as BPB predicts downstream performance, MorphBPE should not be used over BPE. Note that our implementation treats morpheme boundaries the same as whitespace, which may skew results.

6 Conclusion

We presented the first systematic extrinsic evaluation of SuperBPE and MorphBPE against standard BPE across English, Mandarin, and Hungarian using BPB-normalized perplexity at vocabulary sizes of 8K, 16K, and 32K. Two findings emerged.

First, SuperBPE matches BPE for English (within 0.01 BPB) but underperforms for Hungarian (0.01–0.02 BPB) and Mandarin (0.02–0.06 BPB). This suggests that cross-whitespace merging, SuperBPE's defining feature, is counterproductive for languages where whitespace is absent or uncommon.

Second, MorphBPE performs worse than BPE in all settings, with gaps reaching 0.04 BPB at 32K. Even after accounting for the case sensitivity of our implementation, we find no evidence that morphology-aware tokenization improves pretraining efficiency at the 50M-parameter scale. This may be caused by our simplifying treatment of whitespace and morpheme boundaries as identical.

These results carry a cautionary implication: that tokenizer improvements motivated by linguistic theory or intrinsic metrics do not automatically translate into better extrinsic performance. Standard BPE remains a *surprisingly* strong baseline.

Our study is limited by model scale (50M parameters, 50M training tokens), our BPB-based evaluation (which does not perfectly normalize cross-lingual information density), and our language coverage. We hope future work validates these findings at larger scale, against downstream task performance, and across a broader set of languages.

7 Future Work

Several directions merit further investigation. First, validating BPB rankings against concrete downstream tasks across languages would test a core assumption of our methodology. Second, it could be determined whether tokenizer ranking is preserved if our experiments were scaled to larger models (200M–1B parameters), or, following Liu et al. (2025), if BPB rankings change when the three most common tokens are excluded or if inference compute are held constant. Finally, extending this benchmark to Semitic, polysynthetic, and tonal languages would broaden our understanding of how tokenizer design interacts with linguistic diversity.

Beyond evaluation methodology, this work opens up questions about tokenizer design itself. MorphBPE's theoretical advantages for agglutinative languages motivate further development of morphology-aware tokenizers, particularly implementations that faithfully respect the original design's distinction between whitespace and morpheme boundaries. It would also be interesting to explore hybrid approaches, such as adding Su-

perBPE's second training stage to MorphBPE's to allow for efficient cross-morpheme merges.

Limitations

While our evaluation provides the first systematic extrinsic comparison of recent BPE-variant tokenizers across typologically distinct languages, several limitations of our methodology and implementation warrant explicit acknowledgment.

A fundamental conceptual limitation concerns the BPB normalization itself. As Arnett and Bergen (2025) demonstrates, different languages encode different amounts of information per byte. Mandarin, for instance, packs substantially more semantic content per byte than English due to character density, resulting in a higher BPB. As such, BPB scores *cannot* be compared across languages, although this does not affect our core findings.

On the implementation side, our MorphBPE setup deviates from the original paper in that we treat whitespace and morpheme boundaries identically during training, whereas the original design distinguishes between them. This simplification reduces implementation complexity but likely changes MorphBPE's effectiveness slightly.

Finally, we acknowledge that our use of a 50M-parameter model, while a practical and well-motivated proxy for tokenizer effects, may not generalize to larger models. Tokenizer performance is known to scale with model size, but the relative ordering of tokenizers observed at 50M parameters may shift at the scale of models used in production. The degree to which our findings transfer to larger settings remains an open question. Liu et al. (2025) finds that the BPB tokenizer rankings of BPE and SuperBPE switch at high compute levels, except if the three most common tokens in the corpus are excluded or if inference compute is held constant in addition to training.

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David R. Mortensen, Noah A. Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. *Preprint*, arXiv:2305.13707.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Schulze Buschhoff, Charvi Jain, Alexander Arno 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? *arXiv preprint*. ArXiv:2310.08754 [cs].

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

Catherine Arnett, Pamela D. Rivière, Tyler A. Chang, and Sean Trott. 2024. Different tokenization schemes lead to comparable performance in spanish number agreement. *Preprint*, arXiv:2403.13754.

Ehsaneddin Asgari, Yassine El Kheir, and Mohammad Ali Sadraei Javaheri. 2025. MorphBPE: A Morpho-Aware Tokenizer Bridging Linguistic Complexity for Efficient LLM Training Across Morphologies. *arXiv preprint*. ArXiv:2502.00894 [cs].

Khuyagbaatar Batsuren, Gábor Bella, and Fausto Giunchiglia. 2021. MorphyNet: a large multilingual database of derivational and inflectional morphology. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 39–48, Online. Association for Computational Linguistics.

Thomas Bauwens and Pieter Delobelle. 2024. BPE-knockout: Pruning pre-existing BPE tokenisers with backwards-compatible morphological semi-supervision. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5810–5832, Mexico City, Mexico. Association for Computational Linguistics.

Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. 2025. Understanding emergent abilities of language models from the loss perspective. *Preprint*, arXiv:2403.15796.

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

Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. 2024. Unpacking tokenization: Evaluating text compression and its correlation with model performance. *Preprint*, arXiv:2403.06265.

Valentin Hofmann, Janet B. Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Derivational morphology improves bert's interpretation of complex words. *Preprint*, arXiv:2101.00403.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Alisa Liu, Jonathan Hayase, Valentin Hofmann, Sewoong Oh, Noah A. Smith, and Yejin Choi. 2025. SuperBPE: Space Travel for Language Models. *arXiv preprint*. ArXiv:2503.13423 [cs].

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

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

Nishant Luitel, Nirajan Bekoju, Anand Kumar Sah, and Subarna Shakya. 2025. Can perplexity predict finetuning performance? an investigation of tokenization effects on sequential language models for nepali. *Preprint*, arXiv:2404.18071.

OpenAI. 2023. tiktoken: Fast bpe tokeniser for openai models. https://github.com/openai/tiktoken.

Iñigo Parra. 2024. Morphological Typology in BPE Subword Productivity and Language Modeling. *arXiv preprint*. ArXiv:2410.23656 [cs].

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, 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. *Preprint*, arXiv:2406.17557.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. 2025. Fineweb2: One pipeline to scale them all – adapting pre-training data processing to every language. *Preprint*, arXiv:2506.20920.

Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *Preprint*, arXiv:2305.15425.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. *arXiv preprint*. ArXiv:2012.15613 [cs].

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. *arXiv preprint*. ArXiv:1508.07909 [cs].

C E Shannon. Prediction and Entropy of Printed English.

Jessica Taylor and Vie McCoy. 2025. The Weighted Perplexity Benchmark: Tokenizer-Normalized Evaluation for Language Model Comparison.

A Appendix

A.1 Model Architecture and Training

For language model evaluation, we use the GPT-2 architecture with approximately 50 million parameters (Radford et al.). This model size serves as a cost-efficient proxy for downstream performance, as tokenizer quality effects are known to scale with model size. Non-embedding parameters were kept constant, but total parameter count differed between models of different vocabulary sizes. Token count was fixed in training, meaning that, due to the larger parameter count, models with 32K vocabularies were trained with approximately 40% more FLOPs than models with 8K vocabularies.

Models were trained using AdamW with a dynamic batch size based on vocab size to ensure efficient use of VRAM (Loshchilov and Hutter, 2019). The batch size for the models with vocabulary sizes of 8K was 128, and their learning rates were 5×10^{-4}. The learning rate was scaled with the square root of the ratio of the new batch size to the old batch size, so the 32K model had a batch size of 32 and a learning rate of 2.5×10^{-4}. We note that the batch size scaling was more aggressive than necessary, and a reduction of approximately 25% for each doubling of vocabulary size would have been more appropriate.

We plot training loss against tokens seen rather than training steps in Figure 1 because batch size varies across vocabulary sizes. Since a fixed token count represents equivalent amounts of data exposure across all models regardless of batch size, tokens seen provides a fairer comparison of model convergence across different tokenizer vocabulary sizes.

Training proceeded until the model saw 50,000,000 tokens. Validation was performed 6 times throughout training, with 4 evaluations in the middle and 2 evaluations at the start and end of training.

Tables 4 and 5 show the hyperparameters used for training.

Hyperparameter	Value
d_{model}	512
d_{ff}	2048
Layers	8
Heads	8
Context length	512
Weight decay	0.1
β_1, β_2	0.9, 0.95
Gradient clip	1.0
Warmup steps	100
Optimizer	AdamW

Table 4: Fixed LLM hyperparameters, shared across all runs.

Hyperparameter	8K	16K	32K
Batch size	128	64	32
Learning rate	5.00×10^{-4}	3.54×10^{-4}	2.50×10^{-4}
Min. LR	5.00×10^{-5}	3.54×10^{-5}	2.50×10^{-5}

Table 5: Vocabulary-scaled LLM training hyperparameters. Batch size is halved at each vocabulary doubling above 8K to keep the LM-head logit tensor within GPU memory; learning rate scales by $\sqrt{\text{batch\_size}/128}$ to compensate.

A.2 Computational Cost

Tokenizer	Vocab	Time (min)	Mem (GB)
BPE	8K	17.7	39.8
	16K	18.4	24.5
	32K	21.2	16.8
SuperBPE	8K	17.4	39.8
	16K	18.0	24.5
	32K	20.6	16.8
MorphBPE	8K	17.9	39.8
	16K	18.7	24.5
	32K	21.6	16.8

Table 6: LM training time and peak GPU memory, averaged across languages. Memory scales inversely with vocabulary size because smaller vocabulary sizes produce longer token sequences, increasing attention cost. Training cost is near-identical across tokenizer algorithms at the same vocabulary size.

All experiments were run in Google Colab. We used a high-RAM CPU instance to train the tokenizers, and we used a 40GB-A100 instance to train the models. For our particular implementation, the wall-clock time for each training run was around twice as long as the real training time due to initialization, evaluation, dataset downloading, and artifact uploading.

Table 6 summarizes model training time and peak memory.

The time to train the tokenizers themselves was negligible for all tokenizers except for SuperBPE tokenizers, which took significantly longer to train in their inter-word-merging phases.

All MorphBPE models were trained with a case-sensitive morpheme segmentor. This means that, for example, _farmers was tokenized as _farmer and _s, while _Farmers was tokenized as _Far, mer, and _s. To verify that this did not skew the results, we retrained a model with MorphBPE with a case-insensitive tokenizer, and found that its BPB score was identical to three decimal places.

Note the fact that morphemes were tokenized with preceding spaces: the s in _farmers was tokenized as _s. This was an implementation shortcut deemed minor enough to be unlikely to change the bottom line.