Subword Tokenization for Low- and Medium-Resource Languages: A Systematic Evaluation

Jón Daðason[1] , **Hrafn Loftsson**[1]
[1]Department of Computer Science
Reykjavik University, Iceland
{jond, hrafn}@ru.is

Abstract

Subword tokenization is a standard technique for pre-trained language models, mapping text into sequences of tokens from a fixed-size vocabulary. Despite its widespread use, the impact of tokenization algorithms and vocabulary sizes on downstream performance remains underexplored, particularly for low- and medium-resource languages, where suboptimal configurations are difficult to compensate with additional data. Prior studies have mainly focused on high-resource languages, individual tokenization algorithms, or fixed vocabulary sizes, limiting their scope. In this paper, we present a systematic evaluation of subword tokenization strategies in six diverse low- and medium-resource languages: Icelandic, Estonian, Basque, Galician, Nepali, and Tajik. We pretrained monolingual TEAMS-Small models using WordPiece, BPE, and Unigram tokenizers with vocabulary sizes of 16k, 32k, and 64k, and additionally compared byte-level and character-level tokenization for Icelandic. Evaluation on a benchmark of NLP tasks revealed that Unigram tokenization with a 64k vocabulary consistently outperformed other configurations, with vocabulary size having a greater impact on downstream performance than algorithm choice. These gains were task-dependent, with statistically significant improvements for part-of-speech tagging, named entity recognition, and question answering, but not for dependency parsing and summarization. Finally, byte-level tokenization provided no measurable advantage in a monolingual setting, suggesting that its benefits are primarily relevant for multilingual models.

1 Introduction

Subword tokenization addresses the challenge of handling words that a model has never seen before. Unlike word-level tokenizers, which treat each word as an indivisible unit, subword tokenization breaks unfamiliar words into smaller, known components. This approach is particularly beneficial for morphologically rich languages, where corpora often contain millions of unique word forms, many of which appear only once.

A key advantage of subword tokenization is its ability to maintain a fixed-size vocabulary while effectively representing an unlimited number of possible words. By decomposing rare or unknown words into familiar subwords, this method significantly mitigates the data sparsity issue inherent in traditional word-level tokenization. Models learn more efficiently because they encounter each subword more frequently, reducing, if not eliminating, out-of-vocabulary (OOV) words. These benefits have made subword tokenization a standard technique in pre-trained language models (LMs) [Sennrich *et al.*, 2016; Kudo, 2018; Devlin *et al.*, 2019].

The vocabulary size, specified during training, directly influences how the tokenizer segments words. A smaller vocabulary reduces memory requirements, but results in more aggressive word splitting and longer token sequences. Conversely, a larger vocabulary preserves more complete words, but requires more memory. Table 1 illustrates this trade-off using an Icelandic news headline processed with WordPiece tokenizers [Wu *et al.*, 2016] trained on the Icelandic Gigaword Corpus (IGC) [Steingrímsson *et al.*, 2018] at different vocabulary sizes.

Vocab.	Tokenized text
8k	`Se·x·tíu flug·ferð·um afl·ýst`
16k	`Sex·tíu flug·ferðum afl·ýst`
32k	`Sex·tíu flug·ferðum aflýst`
64k	`Sex·tíu flugferðum aflýst`
96k	`Sextíu flugferðum aflýst`

Table 1: Impact of vocabulary size on tokenizing the Icelandic headline *Sextíu flugferðum aflýst* ("Sixty flights canceled") using WordPiece. The dot (·) denotes subword boundaries within words.

Although subword tokenizers aim to minimize the number of splits, they may still encounter OOV words that must be decomposed into the smallest available subwords. This could involve breaking unknown words into individual characters. However, since vocabulary size is limited, not all Unicode characters can be included. When encountering sequences of unseen characters that cannot be broken down into smaller subwords, some tokenizers might replace them with a special unknown token, significantly limiting the model's ability to

capture semantic information. This is especially common for scripts that are absent or underrepresented in the training data. For example, Unicode defines over 100,000 Chinese characters, several thousand of which see regular use. Including all of them in a tokenizer's vocabulary would be extremely inefficient for models not primarily designed to process Chinese text. As a result, many such characters are replaced by the unknown token.

The need for the unknown token can be avoided altogether by representing the minimal vocabulary using individual bytes rather than characters. Any Unicode character can be encoded as a sequence of one to four UTF-8 bytes, each taking one of 256 possible values, forming the smallest possible vocabulary. This approach, popularized by RoBERTa [Liu *et al.*, 2019], has since been widely adopted by models such as GPT-2 and its successors [Radford *et al.*, 2019; Brown *et al.*, 2020; OpenAI, 2024].

In resource-constrained settings, tokenization choices have a more pronounced impact, as there is less data to compensate for suboptimal configurations. This makes tokenization particularly critical for low- to medium-resource languages. Additionally, it is often argued that subword tokenizers should ideally segment words along morphological boundaries, especially in morphologically rich languages. The reasoning is that encoding meaningful morphological units, rather than arbitrary substrings, could help models capture semantic information more effectively [Bostrom and Durrett, 2020]. However, empirical findings on this hypothesis remain inconclusive. Some studies report improvements in specific downstream tasks [Hofmann *et al.*, 2021], while others find that morphology-aware tokenization underperforms compared to traditional subword methods [Toraman *et al.*, 2023; Kaya and Tantuğ, 2024].

In this paper, we pre-train encoder-only TEAMS models [Shen *et al.*, 2021] on the IGC using WordPiece, Byte-Pair Encoding (BPE) [Sennrich *et al.*, 2016], and Unigram [Kudo, 2018] tokenizers with varying vocabulary sizes. Each of these algorithms (see Section 2) can use either characters or bytes as its base units. We evaluate character-level WordPiece, BPE, and Unigram, and additionally compare character-level vs. byte-level BPE.

Finally, we take the tokenizer configuration that achieved the best overall performance on the IGC and use it to train tokenizers for Estonian, Basque, Galician, Nepali, and Tajik. We then pre-train TEAMS-Small models for these languages and compare their downstream performance against models trained with a WordPiece tokenizer using a 32k vocabulary. Our goal is to determine whether the trends observed for Icelandic hold for other low- and medium-resource languages and to determine how different subword tokenization algorithms and vocabulary sizes impact downstream performance.

While previous studies have examined the impact of subword tokenization algorithms and vocabulary sizes, most have focused on individual high-resource languages or have been limited in scope. Various studies have evaluated different tokenization strategies at a fixed vocabulary size or tested varying vocabulary sizes within a single algorithm (see Section 3). In contrast, our work systematically explores the effects of different tokenization strategies on downstream performance across six morphologically rich, low- and medium-resource languages. Our findings provide practical guidance for tokenizer selection in resource-constrained settings.

2 Subword Tokenization Algorithms

This section provides a brief overview of the subword tokenization algorithms used in our experiments.

2.1 Byte-Pair Encoding (BPE)

The BPE algorithm is widely used in models such as RoBERTa, OpenAI's GPT series [Radford *et al.*, 2019; Brown *et al.*, 2020; OpenAI, 2024], and ModernBERT [Warner *et al.*, 2024]. It starts with a vocabulary consisting of the smallest elements in the training corpus, typically either individual characters or bytes. The algorithm iteratively expands the vocabulary by merging the most frequent adjacent subword pair at each step. The process continues until the vocabulary reaches a predefined size.

2.2 WordPiece

WordPiece is used in models such as BERT [Devlin *et al.*, 2019] and ELECTRA [Clark *et al.*, 2020]. While it also builds a vocabulary by iteratively merging subword units, it differs from BPE in its selection criterion. Instead of choosing the most frequent pair, WordPiece selects the pair that maximizes the increase in log-likelihood across the training corpus.

2.3 Unigram

The Unigram algorithm is used in models such as XLNet [Yang *et al.*, 2019], and T5 [Raffel *et al.*, 2020]. Unlike BPE and WordPiece, which progressively build their vocabularies through merging, Unigram follows a subtractive approach. It starts with a large vocabulary containing words and frequent substrings and gradually reduces it to a target size. The tokenizer assigns probabilities to subword sequences using a unigram language model. During training, it evaluates the impact of each subword's removal on the overall corpus likelihood. A percentage of the least impactful subwords is iteratively pruned until the vocabulary reaches the desired size.

3 Related Work

3.1 Tokenization Algorithms

Several studies have compared subword tokenization methods in terms of their impact on downstream performance.

Bostrom and Durrett [2020] examined RoBERTa-Base models pre-trained on English and Japanese corpora using BPE and Unigram tokenizers, both with a vocabulary size of 20k. The Unigram tokenizer consistently outperformed BPE across four downstream tasks. In English question answering (QA), the Unigram model achieved an F_1 score 1.1 points higher than the BPE model, while in Japanese QA, the gap widened to 12.3 points in F_1. The authors attribute this difference to the Unigram tokenizer's ability to better handle languages without explicit word boundaries, such as Japanese, and its greater tendency to align subwords with linguistic morphology. Similarly, Zhang et al. [2020] found that for

English abstractive summarization, Unigram yielded comparable or slightly better ROUGE scores than BPE across four datasets. Ali et al. [2024] compared BPE and Unigram using a 2.6B parameter decoder-only model. In an English monolingual setting, Unigram outperformed BPE on a benchmark of NLP tasks by 0.24 percentage points. For a multilingual model trained on German, French, Italian, Spanish, and English text, BPE achieved an average score 0.17 percentage points higher than Unigram. However, performance varied by language, with Unigram performing better for French and Spanish, and BPE for the other languages.

Toraman et al. [2023] investigated BPE and WordPiece tokenizers for Turkish RoBERTa models. Across six downstream tasks, WordPiece performed similarly to or slightly better than BPE, suggesting that while these methods differ algorithmically, their practical impact on downstream performance may be minimal in some cases.

Beyond these comparisons, several studies have evaluated byte-level tokenization. Qarah and Alsanoosy [2024] pretrained BERT-Base models on an Arabic corpus using WordPiece, Unigram, and byte-level BPE tokenizers, each with a 50k vocabulary. They evaluated the models on seven tasks across 29 datasets, for a total of 36 different dataset-task combinations (with some datasets annotated for multiple tasks). The Unigram model achieved the best performance in 21 out of 36 evaluations, surpassing WordPiece by 0.77 percentage points on average. Despite the fact that byte-level BPE obtained the highest compression ratio (i.e., characters per token), it underperformed against the other two algorithms on most tasks, obtaining an average score 0.95 percentage points lower than WordPiece. Liu et al. [2019] similarly found that RoBERTa models pre-trained with byte-level BPE performed slightly worse on English NLP tasks compared to character-level BPE. Zhang et al. [2022] investigated byte-fallback in a Unigram tokenizer and observed that excessive segmentation into byte units resulted in longer token sequences and, in many cases, degraded translation performance. These findings suggest that while byte-based tokenization ensures full character coverage, this benefit does not necessarily translate into improved downstream performance.

A separate line of work has explored morphological tokenization, which aims to segment words based on their internal structure. Toraman et al. [2023] pre-trained a Turkish RoBERTa model with a morphological tokenizer that split words into morphemes using a morphological analyzer. While this tokenizer performed competitively on some tasks, it generally underperformed compared to BPE and WordPiece. The authors noted that errors introduced by the analyzer may have contributed to these results. Given that accurate analyzers are often unavailable or difficult to develop in low-resource settings, this raises concerns about the feasibility of morphological tokenization in such cases. Kaya and Tantuğ [2024] proposed an alternative morphological approach that extracts word roots and appends suffix tags as subwords. However, for a Turkish BERT-Base model, this method performed similarly or worse than a standard WordPiece tokenizer.

Overall, these studies indicate that Unigram tokenizers often outperform BPE or WordPiece, particularly in languages with complex morphology or ambiguous word boundaries. In contrast, byte-level tokenization tends to produce longer token sequences, which can negatively impact performance. Meanwhile, morphological tokenization remains an open research question, with mixed results suggesting that potential linguistic benefits of morpheme-based segmentation may be offset by practical challenges in implementation and the robustness of existing subword methods.

3.2 Vocabulary Size

Beyond the choice of tokenization algorithm, vocabulary size plays a crucial role in model performance by influencing the balance between vocabulary coverage and memory usage.

Zhang et al. [2020] explored the effect of vocabulary size in English abstractive summarization, comparing Unigram tokenizers with vocabulary sizes ranging from 32k to 256k. They found that ROUGE scores improved up to a vocabulary size of 96k, after which performance began to degrade, suggesting diminishing returns beyond a certain threshold.

For morphologically rich languages, larger vocabularies have shown a similar trend of initial performance gains followed by diminishing improvement. Toraman et al. [2023] evaluated vocabulary sizes for BPE, WordPiece, morphological, and word-level tokenizers in a Turkish RoBERTa model, finding that increasing vocabulary size generally improved downstream performance. However, for BPE and WordPiece, both of which outperformed the other tokenizers, the gains were minimal beyond 66k. The authors attributed this to increased vocabulary coverage and a reduction in OOV tokens. Ali et al. [2024] examined the impact of different vocabulary sizes (33k, 50k, 82k, and 100k) for BPE and Unigram tokenizers on downstream performance. For an English monolingual model, a vocabulary size of 33k yielded the highest average score, while larger vocabularies led to slightly lower performance. In contrast, for a multilingual model covering German, French, Italian, Spanish, and English, the best results were achieved with a vocabulary size of 100k, roughly three times larger than the optimal size for the monolingual model. However, the optimal vocabulary size differed between languages in the multilingual setting, with 33k performing best for German and 82k for English.

A more fine-grained analysis was conducted by Kaya and Tantuğ [2024], who varied vocabulary size in WordPiece tokenization for Turkish NLP tasks using a BERT-Base model. Increasing the vocabulary size from 32k to 64k led to substantial improvements, with named entity recognition (NER) scores improving by 1.6 percentage points and QA scores by up to 6.2 percentage points. However, increasing the vocabulary further to 128k and 256k provided only marginal additional gains. The authors suggested that reducing OOV words improves performance by limiting the overall number of token splits, which in turn reduces the likelihood of splits occurring at arbitrary positions rather than at morphological boundaries, potentially improving the model's ability to capture semantic information effectively.

To summarize, these studies suggest that increasing vocabulary size improves model performance up to a certain point, after which returns diminish. While larger vocabularies reduce OOV words, excessively large vocabularies may intro-

duce inefficiencies without significant downstream benefits.

4 Experimental Setup

4.1 Tokenization

We trained WordPiece and BPE tokenizers using the *tokenizers* library for Python [Moi and Patry, 2023]. For character-level tokenizers, we limited single-character subwords to 1,000 to prevent rare characters, such as emojis and mixed-script Unicode characters, from taking up too much of the vocabulary. We normalized the text by removing all Unicode control characters and split on whitespace characters and punctuation marks.

We trained Unigram tokenizers using the SentencePiece library [Kudo and Richardson, 2018] with settings that roughly matched the configuration we used for the WordPiece and BPE tokenizers. During training, we adjusted the character coverage of the tokenizer to restrict the number of single-character subwords to approximately 1,000, and replaced the default NFKC Unicode normalization with the less aggressive NFC normalization, converting all Unicode characters to their canonical forms. Due to the higher memory usage when training with SentencePiece, we limited the training corpus for Icelandic to approximately 2M documents or 654M tokens, and for Nepali to 800k documents or 348M tokens (owing to the greater number of bytes required to represent characters in the Devanagari script).

For all tokenizers, we maintained original casing and retained all accents, and only used tokens that appeared at least twice for training.

4.2 Pre-Training Settings

TEAMS is a bidirectional, encoder-only version of the ELECTRA architecture [Clark *et al.*, 2020]. We pre-trained TEAMS-Small models using TPU v4-8 accelerators and the TensorFlow Model Garden repository [Yu *et al.*, 2020], following the same settings as in [Shen *et al.*, 2021]. The models consist of 14M parameters each and were pre-trained for 500,000 steps with a maximum sequence length of 512 and a batch size of 256. Pre-training took approximately 18 hours per model to complete. Default pre-training parameters were used for all experiments unless otherwise specified.

4.3 Fine-Tuning Settings

We evaluated the models on six tasks: part-of-speech (POS) tagging, NER, dependency parsing (DP), automatic text summarization (ATS), QA, and topic classification (TC). For fine-tuning, we used the example scripts provided with the Hugging Face Transformers library [Wolf *et al.*, 2020]: `run_ner.py` for POS and NER, `run_qa.py` for QA, and `run_classification.py` for TC. Models were fine-tuned for DP using DiaParser [Attardi *et al.*, 2021], a biaffine dependency parser. For ATS, we used TransformerSum,[1] a Python library implementing the extractive summarization approach described by Liu and Lapata [2019].

To ensure a controlled comparison, we applied the same fine-tuning configuration to all models within each task. Although performing hyperparameter optimization for each model individually would likely yield improved results, we prioritized consistency to enable fair evaluation across models and languages. For POS, NER, DP, and ATS, we adopted the settings used by Dađason and Loftsson [2022], which we found to be robust across all languages. For QA and TC, we performed a hyperparameter search over learning rates in $[3e{-}5, 5e{-}5, 8e{-}5, 1e{-}4]$, batch sizes in $[8, 16, 32]$, and number of epochs in $[5, 10, 20]$, selecting a single configuration that performed well across all languages. The final hyperparameters used in our experiments are summarized in Table 2. Note that for DP, the batch size is token-based and corresponds to the number of tokens processed per optimization step.

Task	Metric	BS	LR	Epochs
POS	Accuracy	16	5e-5	20
NER	F_1 Score	16	5e-5	10
DP	LAS	5000	2e-3	200
ATS	ROUGE-2 Recall	8	2e-5	3
QA	F_1 Score	16	8e-5	5
TC	Accuracy	8	1e-4	10

Table 2: The metric, batch size (BS), learning rate (LR), and number of epochs for fine-tuning TEAMS-Small models on each task.

For datasets with predefined training, validation, and test splits, we report results averaged over 10 runs with different random seeds. Datasets with k-fold splits were evaluated using k-fold cross-validation. For datasets without predefined splits, we applied stratified 10-fold cross-validation, except for datasets containing fewer than 10,000 examples, where we used stratified 5-fold cross-validation with five repetitions to reduce variance and increase statistical power for significance testing.

4.4 Pre-Training Corpora

For pre-training, we used the IGC for Icelandic, the Estonian National Corpus (ENC) [Koppel and Kallas, 2022] for Estonian, and EusCrawl [Artetxe *et al.*, 2022] for Basque. For Galician, Nepali, and Tajik, we used the filtered versions of their subsets in the web-crawled mC4 corpus [Xue *et al.*, 2021], using a Gaussian Mixture Model (GMM) classifier as described in [Dađason and Loftsson, 2024] to filter out low-quality documents. Table 3 provides information on the corpora used in our work.

4.5 Evaluation Datasets

We fine-tuned and evaluated each pre-trained LM on a diverse benchmark of downstream tasks. For Icelandic, we used MIM-GOLD [Loftsson *et al.*, 2010] for POS, MIM-GOLD-NER [Ingólfsdóttir *et al.*, 2020] for NER, IcePaHC-UD [Arnardóttir *et al.*, 2020] for DP, IceSum [Dađason *et al.*, 2021] for ATS, and RUQuAD [Skarphéðinsson *et al.*, 2023] for QA. For Estonian, we used EDT-UD [Muischnek *et al.*, 2016] for POS and DP, and EstNER-New [Sirts, 2023] for NER. For Basque, we used BDT-UD [Aranzabe *et al.*, 2015] for POS and DP, and EIEC [Alegria *et al.*, 2004] for

[1] https://github.com/HHousen/TransformerSum

Language	Corpus	Docs (M)	Tokens (B)
Icelandic	IGC	5.13	1.67
Estonian	ENC	2.14	0.43
Basque	EusCrawl	1.59	0.29
Galician	mC4	4.55	1.21
Nepali	mC4	2.94	0.97
Tajik	mC4	1.28	0.56

Table 3: The number of documents (in millions) and space-delimited tokens (in billions) in each pre-training corpus. Note that the statistics for the IGC and ENC corpora exclude subcorpora with web-crawled content or text shuffled on the sentence or paragraph level.

NER. For Galician, we used CTG [Agerri *et al.*, 2018] for POS, NERC [Agerri *et al.*, 2018] for NER, and Galician-TreeGal-UD [Garcia *et al.*, 2018] for DP. For Nepali, we used NNC-CS [Yadava *et al.*, 2008] for POS, EverestNER [Niraula and Chapagain, 2022] for NER, and SIB-200 [Adelani *et al.*, 2024] for TC. Finally, for Tajik, we used WikiANN [Pan *et al.*, 2017] for NER and SIB-200 for TC. Table 4 provides an overview of the datasets, their associated tasks, and the number of labeled examples.

Language	Task	Dataset	Size
Icelandic	POS	MIM-GOLD	1,000,218
Icelandic	NER	MIM-GOLD-NER	1,000,231
Icelandic	DP	IcePaHC-UD	983,668
Icelandic	ATS	IceSum	1,000
Icelandic	QA	RUQuAD	10,530
Estonian	POS	EDT-UD	437,769
Estonian	NER	EstNER-New	184,638
Estonian	DP	EDT-UD	437,769
Basque	POS	BDT-UD	121,443
Basque	NER	EIEC	59,759
Basque	DP	BDT-UD	121,443
Galician	POS	CTG	1,196,734
Galician	NER	NERC	202,334
Galician	DP	Galician-TreeGal-UD	23,479
Nepali	POS	NNC-CS	1,123,528
Nepali	NER	EverestNER	308,353
Nepali	TC	SIB-200	1,004
Tajik	NER	WikiANN	300
Tajik	TC	SIB-200	1,004

Table 4: Languages, tasks, and datasets in the evaluation benchmark, including dataset size measured in number of labeled examples.

5 Results

In this section, we present the results of our experiments with different tokenization strategies across several low- and medium-resource languages.

5.1 Tokenization Configurations for Icelandic

To assess the impact of different tokenization strategies, we pre-trained TEAMS-Small models on the IGC using WordPiece, BPE, and Unigram tokenizers, each with vocabulary sizes of 16k, 32k, and 64k. Table 5 provides statistics for each tokenizer, including the number of tokens generated from the corpus, and the number of pre-training examples.

Tokenizer	Tokens (B)	Examples (M)	Epochs
WordPiece 16k	2.43	4.78	26.8
WordPiece 32k	2.23	4.38	29.2
WordPiece 64k	2.10	4.13	31.0
BPE 16k	2.38	4.80	26.6
BPE 32k	2.18	4.40	29.1
BPE 64k	2.11	4.14	30.9
Unigram 16k	2.31	4.67	27.4
Unigram 32k	2.13	4.31	29.7
Unigram 64k	2.07	4.07	31.4

Table 5: Statistics for each tokenizer, showing the number of tokens (in billions) generated from the IGC, number of pre-training examples (in millions) generated, and number of epochs at 500k pre-training steps.

Larger vocabularies lead to fewer tokens and a higher compression ratio, which in turn affects the number of input sequences extracted from the corpus for pre-training. For example, a WordPiece tokenizer with a 64k vocabulary can represent the IGC with 4.13M examples (with each example consisting of 512 tokens), which is 13.6% fewer than required with a 16k vocabulary. With a fixed number of training steps, this effectively means that models using tokenizers with larger vocabularies are trained for more epochs. In 500k steps, a TEAMS-Small model will learn from each example 31 times with a WordPiece vocabulary of 64k, but only 26.8 times with a 16k vocabulary. Increasing the vocabulary size may therefore allow the model to learn more effectively during pre-training.

Tokenizer	POS	NER	DP	ATS	QA
WP 16k	96.99	91.01	84.62	72.22	57.32
WP 32k	97.01	91.48	**84.83**	**72.65**	60.03
WP 64k	97.06	**91.97**	**84.86**	72.23	**60.68**
BPE 16k	96.54	90.93	84.50	72.17	57.66
BPE 32k	96.90	91.26	84.73	**72.87**	58.59
BPE 64k	97.01	**91.64**	84.73	**72.76**	**60.32**
UG 16k	97.04	90.80	84.27	72.47	58.62
UG 32k	**97.13**	91.50	84.76	72.24	59.67
UG 64k	**97.17**	**92.13**	**84.84**	**72.60**	**60.62**

Table 6: Downstream performance of TEAMS-Small models pre-trained on the IGC with WordPiece (WP), BPE, and Unigram (UG) tokenizers at various vocabulary sizes. Scores in **bold** are statistically indistinguishable from the best result for each task (paired t-test with Holm-Bonferroni correction; $p < 0.05$).

Table 6 presents the downstream performance of each model across five NLP tasks: POS tagging, NER, DP, ATS, and QA. The results show that Unigram tokenization with a 64k vocabulary achieved the highest average score, slightly outperforming WordPiece and BPE at the same vocabulary size. This aligns with prior research suggesting that the Unigram algorithm has an advantage on some tasks [Bostrom and Durrett, 2020; Zhang *et al.*, 2020;

Qarah and Alsanoosy, 2024]. Additionally, WordPiece performs similarly or slightly better than BPE, consistent with previous findings by Toraman et al. [2023].

Across all tokenization algorithms, larger vocabularies tend to yield better downstream performance, consistent with findings by Toraman et al. [2023] and Kaya and Tantuğ [2024]. The effect is particularly pronounced for NER and QA, where increasing the vocabulary from 32k to 64k improved scores by an average of 0.50 and 1.11 percentage points, respectively. A greater compression ratio allows more characters to fit within each 512-token sequence, effectively increasing the amount of textual context available to the model. Tasks that require a broader context window, such as QA, likely benefit from this effect, as they rely on longer-range dependencies.

Overall, these results suggest that while the Unigram algorithm offers a modest advantage, vocabulary size has a greater impact on downstream performance.

5.2 Byte-Level Tokenization for Icelandic

To evaluate the impact of byte-level tokenization on downstream performance, we pre-trained a TEAMS-Small model on the IGC using a byte-level BPE tokenizer with a 64k vocabulary, comparing it against a character-level tokenizer with the same vocabulary size. Table 7 provides statistics for each tokenizer.

Tokenizer	Tokens (B)	Examples (M)	Epochs
Character-level	2.11	4.14	30.9
Byte-level	2.10	4.13	31.0

Table 7: Statistics for each tokenizer, showing the number of tokens (in billions) generated from the IGC, number of pre-training examples (in millions) generated, and number of epochs at 500k pre-training steps.

There were minimal differences between the two tokenizers in terms of compression ratio and the number of examples generated. This is likely due to the fact that we used a tokenizer with a large vocabulary to process a curated, high-quality monolingual corpus. The character-level tokenizer generated 2.11 billion tokens when processing the IGC, of which only 5,475 were unknown tokens (i.e., sequences of characters that did not exist in its vocabulary). As such, there were negligible benefits from representing unknown characters as bytes. Since the tokenizer has a large vocabulary, a considerable portion of its capacity is allocated to byte-level merges that provide no practical advantage in this setting.

Table 8 presents the downstream performance of the character-level and byte-level tokenizers. The two tokenizers performed nearly identically across all tasks, with no statistically significant differences in downstream performance.

For high-quality monolingual corpora, byte-level tokenization does not appear to offer any advantages. Even in noisier corpora containing a variety of Unicode characters or mixed scripts, representing such noise as bytes rather than unknown tokens may not impact downstream performance, unless the downstream datasets contain similar noise distributions. The primary advantages of byte-level tokenization are more apparent in massively multilingual models, where vocabulary constraints make character-level representation impractical, and in generative models intended to be capable of outputting any Unicode character.

Task	Character-level	Byte-level
POS	97.01%	96.99%
NER	91.64%	91.84%
DP	84.73%	84.32%
ATS	72.76%	72.55%
QA	60.32%	60.64%
Avg	81.29%	81.27%

Table 8: Downstream performance of TEAMS-Small models pre-trained on the IGC with character-level and byte-level BPE with a 64k vocabulary.

5.3 Cross-Linguistic Evaluation

For Icelandic, the Unigram tokenizer with a vocabulary size of 64k obtained the highest overall performance. To determine whether the same trends extend to other languages, we pre-trained TEAMS-Small models using a tokenizer with the same configuration on Estonian, Basque, Galician, Nepali, and Tajik corpora. Table 9 provides statistics for each tokenizer.

Language	Tokenizer	Tokens (B)	Ex. (M)	Epochs
Icelandic	WP 32k	2.23	4.38	29.2
Icelandic	UG 64k	2.07	4.07	31.4
Estonian	WP 32k	0.69	1.33	94.4
Estonian	UG 64k	0.62	1.22	105.0
Basque	WP 32k	0.43	0.85	151.3
Basque	UG 64k	0.40	0.79	161.8
Galician	WP 32k	0.88	1.72	74.4
Galician	UG 64k	0.82	1.61	79.4
Nepali	WP 32k	0.79	1.56	82.0
Nepali	UG 64k	0.74	1.46	87.6
Tajik	WP 32k	0.18	0.36	359.2
Tajik	UG 64k	0.17	0.34	379.3

Table 9: Statistics for the WordPiece (WP) and Unigram (UG) tokenizers for each language, showing the number of tokens (in billions) generated from the pre-training corpus, number of pre-training examples (Ex.; in millions) generated, and number of epochs at 500k pre-training steps.

We observe an improvement in compression ratio across all languages, with the largest gain in Estonian, where the pre-training corpus could be encoded with 10% fewer tokens. For Icelandic, the Unigram tokenizer achieved a compression ratio (i.e., characters per token) of 4.18, substantially higher than the 3.88 obtained by the WordPiece tokenizer. A WordPiece tokenizer with a 64k vocabulary achieved a ratio of 4.12 on the IGC (see Table 5), indicating that much of the observed improvement in compression ratio is likely due to the larger vocabulary. Nevertheless, it seems that the Unigram algorithm may be somewhat more effective at capturing subword structure, particularly in morphologically rich languages. Table 10 presents the downstream performance of each model,

comparing the Unigram tokenizer to a WordPiece tokenizer with a 32k vocabulary.

Language	Task	WP 32k	UG 64k	Err. Red.
Icelandic	POS	97.01	**97.17**	5.35%
Icelandic	NER	91.48	**92.13**	7.63%
Icelandic	DP	84.83	84.84	0.07%
Icelandic	ATS	72.65	72.60	-0.18%
Icelandic	QA	60.03	**60.62**	1.48%
Icelandic	Avg	81.20	81.47	1.45%
Estonian	POS	98.04	98.01	-1.53%
Estonian	NER	78.13	78.63	2.29%
Estonian	DP	89.40	89.40	0.00%
Estonian	Avg	88.52	88.68	1.37%
Basque	POS	97.00	97.03	1.00%
Basque	NER	84.42	84.66	1.54%
Basque	DP	84.45	84.44	-0.06%
Basque	Avg	88.62	88.71	0.76%
Galician	POS	98.92	**99.00**	7.41%
Galician	NER	87.40	**88.34**	7.46%
Galician	DP	**86.59**	85.84	-5.59%
Galician	Avg	90.97	91.06	1.00%
Nepali	POS	96.19	**96.25**	1.57%
Nepali	NER	90.67	**91.48**	8.68%
Nepali	TC	78.82	79.71	4.20%
Nepali	Avg	88.56	89.15	5.13%
Tajik	NER	80.14	**82.08**	9.77%
Tajik	TC	80.20	81.25	5.30%
Tajik	Avg	80.17	81.67	7.54%
All	Avg	86.12	86.50	2.70%

Table 10: Downstream performance of TEAMS-Small models pretrained with a WordPiece (WP) tokenizer with a 32k vocabulary and a Unigram (UG) tokenizer with a 64k vocabulary and relative error reduction from WP 32k to UG 64k. Scores in **bold** indicate statistically significant differences between models (paired t-test; $p < 0.05$).

Overall, the results aligned with our findings for Icelandic. The Unigram tokenizer significantly outperformed WordPiece in eight out of 19 tasks, particularly in NER and TC, where average scores improved by 0.85 and 0.97 percentage points, respectively. The only case where WordPiece significantly outperformed Unigram was DP for Galician, where it achieved a 0.75 percentage point higher score.

In contrast, a larger vocabulary appeared to have minimal or slightly negative impact on DP and POS, indicating that these tasks may not benefit from reduced subword fragmentation or increased context to the same extent as NER and TC. This suggests that while Unigram tokenization is generally advantageous, its benefits are task-dependent.

6 Conclusions

In this paper, we have addressed the question of how different subword tokenization algorithms and vocabulary sizes impact downstream performance for low- and medium-resource languages. Our experiments across six morphologically rich languages have demonstrated that a Unigram tokenizer with a 64k vocabulary consistently outperformed other configurations, particularly for tasks like NER and TC, where scores improved by an average of 0.85 and 0.97 percentage points, respectively. These results align with prior research suggesting that Unigram tokenization better preserves linguistic structure, leading to improved downstream performance [Bostrom and Durrett, 2020], though the magnitude of improvement varied by task and language.

Vocabulary size had a greater impact on downstream performance than the choice of tokenization algorithm, with larger vocabularies consistently yielding better results. However, the benefits were task-dependent, with NER and TC showing substantial improvements, while POS tagging and DP saw minimal gains from increased vocabulary size. Our evaluation of byte-level tokenization found no significant advantages for high-quality monolingual corpora, indicating that such methods may be more beneficial in multilingual settings, where broader character coverage is essential.

Overall, these findings provide practical guidance for pretraining LMs in resource-constrained settings, highlighting that optimizing vocabulary size can yield greater performance gains than switching between tokenization algorithms.

Acknowledgments

This research was supported with Cloud TPUs from Google's TPU Research Cloud (TRC).

References

[Adelani *et al.*, 2024] David Adelani, Hannah Liu, Xiaoyu Shen, Nikita Vassilyev, Jesujoba Alabi, et al. SIB-200: A Simple, Inclusive, and Big Evaluation Dataset for Topic Classification in 200+ Languages and Dialects. In Yvette Graham and Matthew Purver, editors, *Proceedings of EACL 2024*, pages 226–245, St. Julian's, Malta, March 2024. ACL.

[Agerri *et al.*, 2018] Rodrigo Agerri, Xavier Gómez Guinovart, German Rigau, and Miguel Anxo Solla Portela. Developing New Linguistic Resources and Tools for the Galician Language. In *Proceedings of LREC 2018*, Miyazaki, Japan, May 2018. ELRA.

[Alegria *et al.*, 2004] Iñaki Alegria, Olatz Arregi, Irene Balza, Nerea Ezeiza, Izaskun Fernandez, et al. Design and Development of a Named Entity Recognizer for an Agglutinative Language. In *First IJCNLP-04 Workshop on Named Entity Recognition*, 2004.

[Ali *et al.*, 2024] Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, et al. Tokenizer Choice For LLM Training: Negligible or Crucial? In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of ACL: NAACL 2024*, pages 3907–3924, Mexico City, Mexico, June 2024. ACL.

[Aranzabe *et al.*, 2015] Maria Jesús Aranzabe, Aitziber Atutxa, Kepa Bengoetxea, Arantza Diaz de Ilarraza, Iakes Goenaga, et al. Automatic Conversion of the Basque Dependency Treebank to Universal Dependencies. In Markus Dickinsons, Erhard Hinrichs, Agnieszka Patejuk, and Adam Przepiórkowski, editors, *Proceedings of*

TLT14, pages 233–241, Warszawa, Poland, 2015. Institute of Computer Science of the Polish Academy of Sciences.

[Arnardóttir *et al.*, 2020] Þórunn Arnardóttir, Hinrik Hafsteinsson, Einar Freyr Sigurðsson, Kristín Bjarnadóttir, Anton Karl Ingason, et al. A Universal Dependencies Conversion Pipeline for a Penn-format Constituency Treebank. In *Proceedings of UDW 2020*, pages 16–25, Barcelona, Spain (Online), 2020. ACL.

[Artetxe *et al.*, 2022] Mikel Artetxe, Itziar Aldabe, Rodrigo Agerri, Olatz Perez-de Viñaspre, and Aitor Soroa. Does Corpus Quality Really Matter for Low-Resource Languages? In *Proceedings of EMNLP 2022*, pages 7383–7390, Abu Dhabi, United Arab Emirates, December 2022. ACL.

[Attardi *et al.*, 2021] Giuseppe Attardi, Daniele Sartiano, and Maria Simi. Biaffine Dependency and Semantic Graph Parsing for Enhanced Universal Dependencies. In *Proceedings of IWPT 2021*, pages 184–188, Online, August 2021. ACL.

[Bostrom and Durrett, 2020] Kaj Bostrom and Greg Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of ACL: EMNLP 2020*, pages 4617–4624, Online, November 2020. ACL.

[Brown *et al.*, 2020] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, et al. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*, 2020.

[Clark *et al.*, 2020] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *Proceedings of ICLR 2020*, Addis Ababa, Ethiopia, 2020.

[Daðason and Loftsson, 2022] Jón Friðrik Daðason and Hrafn Loftsson. Pre-training and Evaluating Transformer-based Language Models for Icelandic. In *Proceedings of LREC 2022*, pages 7386–7391, Marseille, France, June 2022. ELRA.

[Daðason and Loftsson, 2024] Jón Daðason and Hrafn Loftsson. Unsupervised Outlier Detection for Language-Independent Text Quality Filtering. In Maite Melero, Sakriani Sakti, and Claudia Soria, editors, *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages @ LREC-COLING 2024*, pages 383–393, Torino, Italia, May 2024. ELRA and ICCL.

[Daðason *et al.*, 2021] Jón Daðason, Hrafn Loftsson, Salome Sigurðardóttir, and Þorsteinn Björnsson. IceSum: An Icelandic Text Summarization Corpus. In *Proceedings of NAACL-HLT Student Research Workshop 2021*, pages 9–14, Online, June 2021. ACL.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT 2019: Volume 1 (Long and Short Papers*, pages 4171–4186, Minneapolis, Minnesota, June 2019. ACL.

[Garcia *et al.*, 2018] Marcos Garcia, Carlos Gómez-Rodríguez, and Miguel A. Alonso. New Treebank or Repurposed? On the Feasibility of Cross-Lingual Parsing of Romance languages with Universal Dependencies. *Natural Language Engineering*, 24(1):91–122, 01 2018.

[Hofmann *et al.*, 2021] Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. Superbizarre Is Not Superb: Derivational Morphology Improves BERT's Interpretation of Complex Words. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of ACL-IJCNLP 2021*, pages 3594–3608, Online, August 2021. ACL.

[Ingólfsdóttir *et al.*, 2020] Svanhvít L. Ingólfsdóttir, Ásmundur A. Guðjónsson, and Hrafn Loftsson. Named Entity Recognition for Icelandic: Annotated Corpus and Models. In Luis Espinosa-Anke, Carlos Martín-Vide, and Irena Spasić, editors, *Proceedings of SLSP 2020*, pages 46–57, Cardiff, United Kingdom, 2020.

[Kaya and Tantuğ, 2024] Yiğit Bekir Kaya and A. Cüneyd Tantuğ. Effect of tokenization granularity for Turkish large language models. *Intelligent Systems with Applications*, 21:200335, 2024.

[Koppel and Kallas, 2022] Kristina Koppel and Jelena Kallas. Eesti keele ühendkorpuste sari 2013–2021: mahukaim eestikeelsete digitekstide kogu. *Eesti Rakenduslingvistika Ühingu aastaraamat*, 18:207–228, 2022.

[Kudo and Richardson, 2018] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of EMNLP 2018: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. ACL.

[Kudo, 2018] Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of ACL 2018*, pages 66–75, Melbourne, Australia, 2018.

[Liu and Lapata, 2019] Yang Liu and Mirella Lapata. Text Summarization with Pretrained Encoders. In *Proceedings of EMNLP-IJCNLP 2019*, pages 3730–3740, Hong Kong, China, November 2019. ACL.

[Liu *et al.*, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

[Loftsson *et al.*, 2010] Hrafn Loftsson, Jökull H. Yngvason, Sigrún Helgadóttir, and Eiríkur Rögnvaldsson. Developing a PoS-tagged corpus using existing tools. In Francis M. Tyers Sarasola, Kepa and Mikel L. Forcada, editors, *Proceedings of 7th SaLTMiL Workshop on Creation and Use of Basic Lexical Resources for Less-Resourced Languages*, LREC 2010, Valetta, Malta, 2010.

[Moi and Patry, 2023] Anthony Moi and Nicolas Patry. HuggingFace's Tokenizers, April 2023.

[Muischnek *et al.*, 2016] Kadri Muischnek, Kaili Müürisep, and Tiina Puolakainen. Estonian Dependency Treebank: from Constraint Grammar tagset to Universal Dependencies. In *Proceedings of LREC 2016*, pages 1558–1565, Portorož, Slovenia, May 2016. ELRA.

[Niraula and Chapagain, 2022] Nobal Niraula and Jeevan Chapagain. Named Entity Recognition for Nepali: Data Sets and Algorithms. *The International FLAIRS Conference Proceedings*, 35, May 2022.

[OpenAI, 2024] OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2302.10198*, 2024.

[Pan *et al.*, 2017] Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. Cross-lingual Name Tagging and Linking for 282 Languages. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of ACL 2017*, pages 1946–1958, Vancouver, Canada, July 2017. ACL.

[Qarah and Alsanoosy, 2024] Faisal Qarah and Tawfeeq Alsanoosy. A Comprehensive Analysis of Various Tokenizers for Arabic Large Language Models. *Applied Sciences*, 14(13), 2024.

[Radford *et al.*, 2019] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners, 2019.

[Raffel *et al.*, 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[Sennrich *et al.*, 2016] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of ACL 2016*, pages 1715–1725, Berlin, Germany, 2016.

[Shen *et al.*, 2021] Jiaming Shen, Jialu Liu, Tianqi Liu, Cong Yu, and Jiawei Han. Training ELECTRA Augmented with Multi-word Selection. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of ACL: ACL-IJCNLP 2021*, pages 2475–2486, Online, August 2021. ACL.

[Sirts, 2023] Kairit Sirts. Estonian Named Entity Recognition: New Datasets and Models. In Tanel Alumäe and Mark Fishel, editors, *Proceedings of NoDaLiDa 2023*, pages 752–761, Tórshavn, Faroe Islands, May 2023. University of Tartu Library.

[Skarphéðinsson *et al.*, 2023] Njáll Skarphéðinsson, Breki Guðmundsson, Steinar Þ. Smári, Marta K. Lárusdóttir, Hafsteinn Einarsson, et al. GameQA: Gamified Mobile App Platform for Building Multiple-Domain Question-Answering Datasets. In Danilo Croce and Luca Soldaini, editors, *Proceedings of EACL 2023: Systems Demonstrations*, pages 152–160, Dubrovnik, Croatia, May 2023. ACL.

[Steingrímsson *et al.*, 2018] Steinþór Steingrímsson, Sigrún Helgadóttir, Eiríkur Rögnvaldsson, Starkaður Barkarson, and Jón Guðnason. Risamálheild: A Very Large Icelandic Text Corpus. In *Proceedings of LREC 2018*, Miyazaki, Japan, May 2018. ELRA.

[Toraman *et al.*, 2023] Cagri Toraman, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozcelik. Impact of Tokenization on Language Models: An Analysis for Turkish. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(4), March 2023.

[Warner *et al.*, 2024] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, et al. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. *arXiv preprint arXiv:2412.13663*, 2024.

[Wolf *et al.*, 2020] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, et al. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of EMNLP 2020: System Demonstrations*, pages 38–45, Online, October 2020. ACL.

[Wu *et al.*, 2016] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*, 2016.

[Xue *et al.*, 2021] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, et al. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In *Proceedings of NAACL-HLT 2021"*, pages 483–498, Online, June 2021. ACL.

[Yadava *et al.*, 2008] Yogendra P. Yadava, Andrew Hardie, Ram Raj Lohani, Bhim N. Regmi, Srishtee Gurung, et al. Construction and annotation of a corpus of contemporary Nepali. *Corpora*, 3(2):213–225, 2008.

[Yang *et al.*, 2019] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[Yu *et al.*, 2020] Hongkun Yu, Chen Chen, Xianzhi Du, Yeqing Li, Abdullah Rashwan, et al. TensorFlow Model Garden, 2020.

[Zhang *et al.*, 2020] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR, July 2020.

[Zhang *et al.*, 2022] Shiyue Zhang, Vishrav Chaudhary, Naman Goyal, James Cross, Guillaume Wenzek, et al. How Robust is Neural Machine Translation to Language Imbalance in Multilingual Tokenizer Training? In Kevin Duh and Francisco Guzmán, editors, *Proceedings of AMTA 2022*, pages 97–116, Orlando, USA, September 2022. Association for Machine Translation in the Americas.