Unpacking Tokenization: Evaluating Text Compression and its Correlation with Model Performance

Omer Goldman$^{\beta\gamma}$ **Avi Caciularu**$^{\gamma}$ **Matan Eyal**$^{\gamma}$
Kris Cao$^{\delta}$ **Idan Szpektor**$^{\gamma}$ **Reut Tsarfaty**$^{\gamma}$

$^{\beta}$Bar-Ilan University $^{\gamma}$Google Research $^{\delta}$Google DeepMind
{ogoldman,avica,matane,kriscao,szpektor,reutt}@google.com

Abstract

Despite it being the cornerstone of BPE, the most common tokenization algorithm, the importance of compression in the tokenization process is still unclear. In this paper, we argue for the theoretical importance of compression, that can be viewed as 0-gram language modeling where equal probability is assigned to all tokens.We also demonstrate the empirical importance of compression for downstream success of pre-trained language models. We control the compression ability of several BPE tokenizers by varying the amount of documents available during their training: from 1 million documents to a character-based tokenizer equivalent to no training data at all. We then pre-train English language models based on those tokenizers and fine-tune them over several tasks. We show that there is a correlation between tokenizers' compression and models' downstream performance, suggesting that compression is a reliable intrinsic indicator of tokenization quality. These correlations are more pronounced for generation tasks (over classification) or for smaller models (over large ones). We replicated a representative part of our experiments on Turkish and found similar results, confirming that our results hold for languages with typological characteristics dissimilar to English. We conclude that building better compressing tokenizers is a fruitful avenue for further research and for improving overall model performance.

1 Introduction

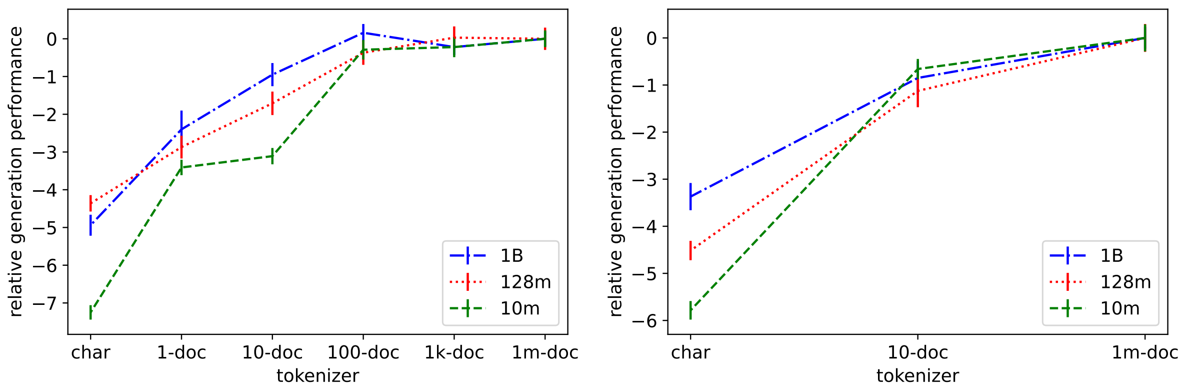
While language modeling pipelines employ a multitude of sophisticated techniques to achieve success in many NLP tasks, their presupposed tokenization, i.e., the step of discretizing text into processable units, is often done with less scrutiny or deviation from the common practices. This tokenization stage, which segments space-delimited words into subwords, forms the foundation of most large language models (LLMs; Touvron et al., 2023; Gemini, 2023; Groeneveld et al., 2024, inter alia) and influences their modus operandi in subsequent usage. Among other open questions regarding tokenization, it is unclear whether tokenization is even needed (Clark et al., 2022; Xue et al., 2022; Keren et al., 2022) and how much poor tokenization influences model performance, especially for non-English languages (Klein and Tsarfaty, 2020; Rust et al., 2021; Gueta et al., 2023).

As the tokenizers serve language models, it is straightforward that the primary method to assess their quality is by measuring their contribution to the model performance over the NLP tasks it is meant to solve, i.e., evaluating the tokenizers on tasks extrinsic to tokenization itself. However, this method requires pretraing expensive LLMs whenever an evaluation of a tokenizer is needed. For this reason an intrisic indicator of tokenization quality is warrented. And indeed the literature is teeming with intrinsic evaluations of tokenization. For example, Sennrich et al. (2016) used text compression as the main indicator of the tokenizer's intrinsic quality, whereas Bostrom and Durrett (2020) suggested assessing tokenizers based on segmentation overlap with a morphologically segmented reference.

In this paper we carefully distinguish between intrinsic and extrinsic evaluation of tokenizers, and examine to what extent they are correlated. As a specific intrinsic evaluation we focus on compression, the metric underpinning BPE (Sennrich et al., 2016), the most prevalent tokenization algorithm that requires character co-occurrence statistics over a large corpus of raw text to achieve minimal length in tokens. We control the tokenizer's ability to compress by limiting its *support*, i.e., the amount of data available in the tokenizer's training corpus. By doing so we skew the statistics available to the tokenizer.

We compared tokenizers trained with a million supporting documents to ones trained on less and

Findings of the Association for Computational Linguistics ACL 2024, pages 2274–2286
August 11-16, 2024 ©2024 Association for Computational Linguistics



(a) English

(b) Turkish

Figure 1: Generation performance of the various models averaged over both generation tasks. For each model size the results are presented as relative compared to the 1M-DOC model.

less data, down to a single document, and to a character-level tokenizer, equivalent to *zero support*. We then pre-trained from scratch copies of a decoder-only transformer-based model (Vaswani et al., 2017), with the different tokenizers, and fine-tuned them on several downstream tasks. In this work we hypothesize that the downstream success should be correlated with the compression ability of the underlying tokenizers. We experimented with three model sizes, tokenizers of six different volumes of supporting data, and two languages, English and Turkish.

Our results show that in terms of intrinsic performance, the tokenizers' compression ability is highly influenced by the amount of supporting data, with tokenizers trained on a minimal amount of data having tokenized texts more than 60% longer compared to the best compressing tokenizer. However, the discrepancy in compression is significantly more marked for less frequent words.

Extrinsically, we also found that downstream success monotonically increases with the increase in the tokenizer's support. The correlation between the intrinsic and extrinsic measures of tokenization quality points to the conclusion that better compressing tokenizers is a desired goal on the road to better language models. A conclusion that may be true even for models dealing with other modalities (Ryoo et al., 2021; Ronen et al., 2023).

While we evaluated the downstream performance on both classification and generative tasks, we observed that the correlation to compression is stronger for the latter type of tasks. This discrepancy could be attributed to the fact that generative tasks require the use of the tokenizer more extensively than in classification tasks, aligning with the number of generation steps involved. We therefore conclude that tokenization's effect is better assessed through generation tasks, rather than classification tasks.

Our results also show that smaller models are especially vulnerable to poor tokenizations, with the smallest $10m$ parameter model suffering from more significant drops in performance compared to our largest $1B$ model. Finally, experimentation with Turkish revealed the same trends, ruling out the option of an English-specific phenomenon.

In the remainder of the paper we will describe the common practices in assessing tokenizers (section 2) and argue for the theoretical sensibility of compression as an intrinsic tokenization evaluation (section 3). We will then describe our experiments (section 4) and their results (section 5).

2 Measuring Tokenization Quality

From the very early days of NLP, models have always assumed text discretized to tokens as input (Winograd, 1971). For the most part, these tokens were whitespace separated words, but in the recent decade non-trivial tokenization algorithms have surfaced (Mikolov et al., 2012; Sennrich et al., 2016; Ataman and Federico, 2018), primarily to deal with unseen tokens without smoothing techniques or other convoluted methods (Chen and Goodman, 1999). The underlying reasoning behind all modern tokenization methods is that some subwords, e.g., morphemes, may carry independent information that is of value to the model even if the word as a whole is rare or unseen. Therefore, better tokenization is assumed to improve models' performance

TOKENIZER	SENTENCE
CHAR	_T h i s _i s _a b o u t _c o m p r e s s i n g _t o k e n i z e r s
1-DOC	_Th is _is _a b ou t _comp re ss ing _to k en i z ers
10-DOC	_This _is _about _comp res sing _to k en iz ers
100-DOC	_This _is _about _comp ress ing _tok en izers
1K-DOC	_This _is _about _comp ressing _to ken izers
1M-DOC	_This _is _about _compress ing _token izers

Figure 2: Six tokenizers differing in the amount of supporting documents tokenizing the same sentence. Note that better compression is achieved with more support.

over rare words, while also carrying computational benefits, like smaller models and the elimination of unknown tokens.

It is not surprising then, that whenever tokenizers are presented or tested, they are usually accompanied with an array of evaluations that assess the tokenization's influence on the model's downstream success, mostly on translation tasks (Kudo, 2018; Provilkov et al., 2020; Vilar and Federico, 2021; Saleva and Lignos, 2023), although monolingual tasks are also used (Yehezkel and Pinter, 2023).

Other works circle back and assess tokenization with respect to the desiderata it is supposed to serve as a stand-alone algorithm, independently from the model trained on top of it. This is done usually in addition to evaluation over downstream performance. However, most works disagree on the desiderata themselves. Many emphasize alignment to linguistically meaningful units (Klein and Tsarfaty, 2020; Hofmann et al., 2021, 2022; Gow-Smith et al., 2022) or to human cognitive preferences in general (Beinborn and Pinter, 2023).[1] Others include analyses of token length and frequency (Bostrom and Durrett, 2020; Yehezkel and Pinter, 2023), mostly in addition to the above, assuming that ideal tokenizers use longer and more frequent tokens.

The two types of tokenization evaluations, extrinsic over downstream success and intrinsic over a plethora of metrics, are usually not compared directly. They are only used to demonstrate the superiority of a specific tokenizer, and the relations between the *evaluation approaches* is glossed over. In this work we explicitly focus on compression as a potential intrinsic indicator of tokenization quality, as has been suggested in past works in other settings (Gallé, 2019; Gutierrez-Vasques et al., 2023), and check to what extent it is correlated with extrinsic downstream success. We conclude that compression is a desideratum for tokenization not only due to its theoretical virtue, expanded on in the next section, but first and foremost because it correlates with downstream performance.

3 The Role of Compression in Tokenization

In the realm of intrinsic measures for evaluating tokenization quality, compression particularly stands out in prominence. It has garnered considerable attention, notably due to its pivotal role as the cornerstone of the byte pair encoding tokenization algorithm (BPE; Schuster and Nakajima, 2012; Sennrich et al., 2016), an algorithm that initially conceived for general data compression purposes (Gage, 1994).

Given data composed of sequences of atomic symbols, the algorithm minimizes the overall data length, for a given dictionary budget, by iteratively substituting a new symbol in place of the symbol pair most frequently occurring in a large corpus of *supporting* data. In the domain of language modeling, the symbols are usually characters and the supporting corpus is a subset of the text designated to be used as a training set for the language model which the tokenizer is meant to serve.

But in a sense, compression-driven tokenization may be viewed as language modeling in and of itself. Consider that language models are aimed at assessing and possibly maximizing the likelihood of produced texts expressed as a product of token probabilities,

$$P(\mathbf{x}) = \prod_{k} P(x_k|\mathbf{x}_{1:k-1}),$$

where x_k is the kth token in a sentence $\mathbf{x}$. Compression limits the lower bound on this product of

[1] This line of works may view tokenization as a continuation of unsupervised morphemic segmentation (Creutz and Lagus, 2002; Virpioja et al., 2013).

fractions by minimizing the number of operands, i.e., minimizing the length of the sequence.

In terms of n-gram language modeling, where the probability of each token is approximated as dependent only on a context of length $n-1$,

$$P(\mathbf{x}) \approx \prod_k P(x_k|\mathbf{x}_{k-(n-1):k-1}),$$

and the probability of x_k given its context is further approximated by the number of appearances of the relevant n-gram in a training corpus,

$$P(x_k|\mathbf{x}_{k-(n-1):k-1}) \propto N(\mathbf{x}_{k-(n-1):k}),$$

A compressor may be considered a 0-gram language model, where the relevant n-gram is of length 0 and the probability of each token is not even a function of its own frequency in the training data, setting uniformly,

$$P(x_i) = |V|^{-1},$$

where $|V|$ is the vocabulary size.

Although simplistic when thinking about language modeling with predefined whitespace-separated words, this type of objective is sensible when considering that it is used to determine the symbols themselves.

From this point of view, prioritizing compression as an indicator for tokenization quality is very reasonable. Since BPE optimizes an approximation, albeit crude, of the downstream objective, doing better under this approximated objective should translate into better downstream performance which will justify the focus on compression as a metric.

Moreover, from an information theoretic perspective, Shannon's source coding theorem (Shannon, 1948) links the limit on the compression to the entropy of the source of the data to be compressed. As language models aim to increase the log-likelihood of texts, hence decrease the entropy of the distribution, they inadvertently also increase the possible compression of the texts. Our claim is that this relationship is symmetric, and BPE tokenizers, as they compress texts, may also inadvertently increase their log-likelihood.

We set to empirically examine our hypothesis by assessing the correlation between the tokenizer's compression ability and the performance of language models of various sizes over a set of downstream tasks.

To explicitly control the compression ability, while fixing any other intervening factors as much as possible, we deal only with BPE tokenizers. This is in contrast with other works that compared compression across different tokenization algorithms (Gallé, 2019; Schmidt et al., 2024). To create BPE tokenizers with varied compression rate we recall that BPE's maximal compression is guaranteed only over its supporting corpus. Normally, for large enough corpora, a minimal discrepancy is assumed between the character distribution in the corpus and the "true" distribution in the language. In this work however, we explicitly emphasize and expand this discrepancy by limiting the size of the support to a great extent. We will show that this intervention severely hinders the compression capacity of the tokenizer and that it also leads to deteriorating downstream performance.

4 Experimental Setup

4.1 English Experiments

Tokenizers We trained six different tokenizers with dictionary size of up to $32k$ tokens.[2] Each tokenizer was supported by a different amount of documents from the model's train set: a million (1M-DOC), a thousand (1K-DOC), a hundred (100-DOC), ten (10-DOC), one document (1-DOC) and no documents at all (CHAR). The tokenizers are initialized with all the relevant symbols - the characters of the alphabet, punctuation marks, and all foreign characters that appear on the respective documents.

Models For every tokenizer, we trained decoder-only transformer-based language models in three sizes, in terms of number of parameters: *1B*, *128m* and *10m*. The model sizes exclude the parameters dedicated to the embedding layer, as its size may differ across tokenizers. See Appendix A for further details.

Data Pretraining of the English models was executed monolingually using the train split of C4 (Raffel et al., 2020).

Tasks To evaluate the tokenizers' contribution to downstream success we finetuned the models over four tasks. Two classification tasks:

[2] For some tokenizers with little supporting data, there were less than $32k$ strings and sub-strings so the vocabulary in practice was smaller. See Appendix A for details.

Tokenizer	Token Length	Relative Length
1M-DOC	9,336,052	—
1K-DOC	9,541,368	+2%
100-DOC	10,489,029	+12%
10-DOC	15,126,769	+62%
1-DOC	20,647,861	+121%
CHAR	39,480,577	+323%

Table 1: Compression ability of the different tokenizer, accumulative over the development sets of all downstream tasks. Relative length is in comparison to the 1M-DOC tokenizer.

- *QQP* (Quora Question Pairs[3]) where the task is to classify 2 questions as duplicates or not.
- *MultiNLI* (Williams et al., 2018) where the model is tested on natural language inference (NLI) examples from a domain which differs from the ones appearing at the training set.

And two generation tasks:

- *X-Sum* (Narayan et al., 2018) where news articles should be summarized to one single sentence.
- *QG-QA* (Question Generation over SQuAD; Rajpurkar et al., 2016) where the task is to generate questions based on a context paragraph and an answer.

4.2 Turkish Experiments

To make sure that our results are not due to English-specific phenomena, we repeat a representative subest of the experiments with Turkish, an agglutinative language with higher morphemes-per-word ratio. for the purpose of intrinsic evaluation, we trained six Turkish tokenizers, as we did for English. However for extrinsic evaluation, due to the expensive pretraining and finetuning, we trained models only for three tokenizers: 1M-DOC, 10-DOC, and CHAR. The models were pretrained on the train split of the Turkish part of mC4 (Xue et al., 2021), and finetuned over three tasks: one classification task, XNLI (Conneau et al., 2018), and two generation tasks, XL-Sum (Hasan et al., 2021) and Question Generation over the TQuAD dataset.[4]

5 Results

5.1 Intrinsic Evaluation

To illustrate the effect of limiting the tokenization support on the compression ability, we measured the accumulative length in tokens of the development sets of all English downstream tasks.

The results, depicted in Table 1, show that providing less support severely impedes the tokenizer ability to compress unseen texts. Note that the inflation in texts' length is not linear. Reducing the supporting data amount by three orders of magnitude, from 1M-DOC to 1K-DOC, results in only slightly longer texts, while a reduction in another three orders of magnitude to the 1-DOC tokenizer carries an effect much more significant.

5.2 Extrinsic Evaluation

Table 2 summarizes the downstream evaluation results for all models and all tokenizers over all four English tasks. Unsurprisingly, it shows that larger models, in terms of parameters, fare better on all tasks. Additionally, it shows that all models perform better on the classification tasks compared to the generation tasks. Nevertheless, over most tasks and model sizes, there is a clear improvement in performance when the models are equipped with better supported tokenizers.

Similarly to the intrinsic metric above, the downstream improvement is not linear as well. The improvements achieved by updating the tokenizer from the 1-DOC to the 10-DOC are more substantial than those from 1K-DOC to 1M-DOC, despite the introduction of significantly fewer documents.

The findings for the Turkish models in Table 5 demonstrate analogous patterns, indicating that the results decline as the tokenizer's support diminishes. This is again particularly noticeable in the case of generation tasks.

5.3 Intrinsic-Extrinsic Correlation

To assess the correlation between the tokenizer's support and the model's task performance we computed the Spearman's ρ coefficient, separately for each task and each model size. This correlation coefficient was chosen since it refers to the relative rank of each data point, thus it does not ascribe linear importance to the differences in the absolute number of supporting documents.

The results are shown in Table 3. Note that due to the small sample size the correlation is statistically significant ($\alpha = 0.05$) only for coefficients

[3] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

[4] https://tquad.github.io/turkish-nlp-qa-dataset/

Tokenizer	Task			
	QQP (F1)	MultiNLI (Acc.)	XSum (RougeL)	QG-QA (RougeL)
		1B params		
1M-DOC	88.02±0.18	88.24±0.10	47.71±0.02	33.09±0.41
1K-DOC	87.38±0.05	88.32±0.07	47.53±0.04	32.95±0.42
100-DOC	88.30±0.09	88.75±0.11	47.69±0.05	32.99±0.46
10-DOC	87.44±0.07	88.27±0.09	47.07±0.06	30.51±0.61
1-DOC	86.07±0.20	86.67±0.25	46.33±0.11	28.42±0.99
CHAR	83.13±0.23	84.59±0.65	44.69±0.08	24.91±0.55
		128m params		
1M-DOC	82.13±0.14	85.33±0.16	45.68±0.04	27.66±0.59
1K-DOC	82.29±0.06	85.45±0.18	45.83±0.08	27.37±0.59
100-DOC	81.75±0.28	85.07±0.13	45.53±0.02	26.97±0.64
10-DOC	80.14±0.19	83.76±0.06	45.08±0.04	24.99±0.62
1-DOC	78.71±0.19	82.30±0.31	44.43±0.06	23.9±0.60
CHAR	76.27±0.23	82.10±0.26	43.19±0.06	21.81±0.43
		10m params		
1M-DOC	71.65±0.81	78.62±0.22	40.92±0.06	22.43±0.44
1K-DOC	69.97±0.11	79.94±0.15	40.69±0.08	22.13±0.53
100-DOC	71.26±0.28	78.57±0.17	41.05±0.02	21.59±0.52
10-DOC	66.51±0.22	75.95±0.24	39.00±0.04	19.73±0.43
1-DOC	66.25±0.11	78.11±0.12	37.01±0.08	18.61±0.40
CHAR	64.01±0.14	75.81±0.77	27.86±0.04	16.92±0.38

Table 2: Results over all downstream tasks, each in terms of its respective metric. Results are averaged over 5 finetues.

Model Size	Task			
	QQP	MultiNLI	XSum	QG-QA
1B	0.714	0.600	0.943**	0.943**
128m	0.943**	0.943**	0.943**	1.000**
10m	0.943**	0.886*	0.829*	1.000**

Table 3: Spearman's ρ coefficient for rank correlation between downstream performance and the tokenizers' support for all model sizes and tasks. Asterisks are used to denote statistically significant correlation. * $p < 0.05$, ** $p < 0.01$

Model Size	Task			
	QQP	MultiNLI	XSum	QG-QA
1B	-0.980**	-0.974**	-0.994**	-0.976**
128m	-0.971**	-0.863*	-0.988**	-0.949**
10m	-0.870*	-0.710	-0.996**	-0.933**

Table 4: Pearson's correlation coefficient between downstream performance and the tokenizers' compression as expressed by the dev sets length (see Table 1), for all model sizes and tasks. Asterisks are used to denote statistically significant correlation. * $p < 0.05$, ** $p < 0.01$

larger than 0.829. The results show that, for the most part, the tokenizer's support is well correlated with the model's overall success, with the clear exception of classification tasks on the 1B model.

Even starker correlation appears when measuring the Pearson's correlation coefficient between downstream performance and the compression itself, i.e., to the overall length of the development sets in tokens, from Table 1. As can be seen in Table 4, the inverse correlation between length in tokens and performance is high, with the exception of classification tasks on the 10m model. Note that here as well, the small sample size causes the correlation to be statistically significant only when above 0.729 in absolute value. No numerical correlation could be computed for the Turkish results due to the small sample size.

These results point to generation tasks as better downstream evaluators of tokenizers, as tokenization is less crucial to the models' success over

Tokenizer	Task		
	XNLI	XLSum	QG-QA
	1B params		
1m-doc	79.56	46.48	29.78
10-doc	79.28	45.92	29.02
char	75.18	40.00	24.69
	128m params		
1m-doc	76.23	45.07	27.45
10-doc	74.83	43.40	26.87
char	68.88	40.33	23.16
	10m params		
1m-doc	63.60	37.16	21.76
10-doc	64.06	37.27	19.95
char	63.73	35.56	16.62

Table 5: Results for the Turkish models over all downstream tasks, each in terms of its respective metric: accuracy for XNLI, and RougeL for XL-Sum and QG-QA.

classification tasks.

In addition to assessing the correlation's significance, Figure 1 visualizes the effect size for both languages. We averaged the performance over the generation tasks, as the correlation was less significant for classification. The graph depicts, separately for each model size, the performance with the various tokenizers compared to the best supported tokenizer. Notably, while compression consistently correlates with generation performance across all model sizes, the impact is particularly pronounced for smaller models.

The parallels drawn between tokenization and language modeling in section 3 may provide some explanation to the smaller effect of poorer tokenization on larger models. As we claim that compression is simple language modeling on its own, it is possible that LLMs that are more powerful language models in general are able to allocate resources to compensate for less compressing tokenizers.

6 Analysis

Tokenization of Frequent Words To better understand the source for discrepancy in compression between tokenizers, we plot in Figure 3 the number of tokens needed per word with respect to the word frequency (measured in number of appearances in a sample of 3 million unseen documents from mC4). We averaged the token-per-word ratio over all words whose occurrences are of the same order of magnitude and provided the number of words in each bin. A similar analysis was done for Turkish and it is shown in Figure 4.

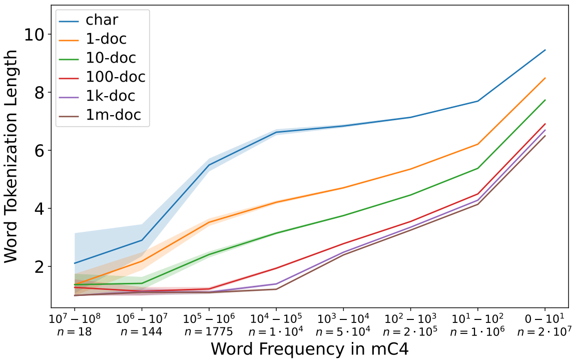


Figure 3: Number of subwords per *English* word as a function of its abundance in 3 million unseen documents. Averaged over orders of magnitude. The number words included in each bin is indicated under the x axis.

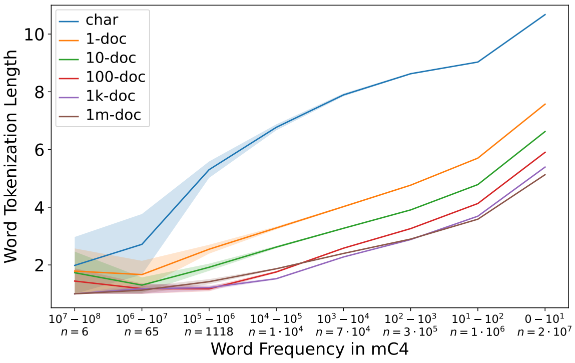
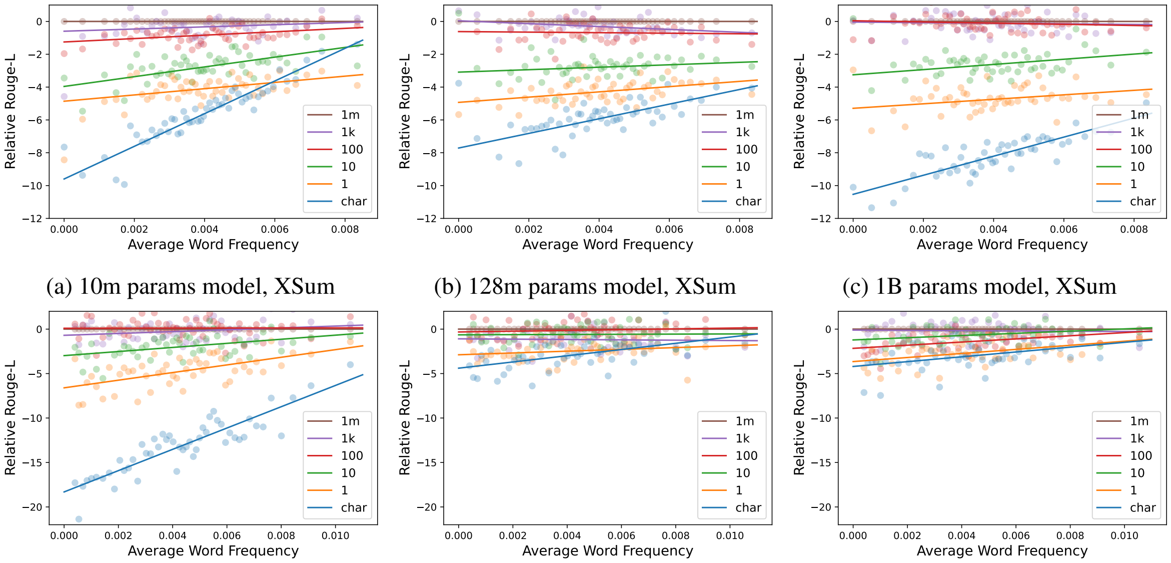


Figure 4: Number of subwords per *Turkish* word as a function of its abundance in 3 million unseen documents. Averaged over orders of magnitude. The number words included in each bin is indicated under the x axis.

The figures show that the token-to-word ratio is extremely similar across tokenizers for words that are the most frequent. On the other hand, the different tokenizers diverge in token-to-word ratio when presented with rarer words, with less supported tokenizers being more sensitive to word frequency, compared to better supported tokenizers. It is worth noting that the same trend applies to the char tokenizer, for which the number of tokens per word is simply its length in characters. This should not be surprising due to the tendency of frequently used words to be shorter in accordance to Zipf's law of abbreviation (Zipf, 1949).

In addition, as predicted by Zipf's law (Estoup,



(a) 10m params model, XSum

(b) 128m params model, XSum

(c) 1B params model, XSum

(d) 10m params model, QG-QA

(e) 128m params model, QG-QA

(f) 1B params model, QG-QA

Figure 5: Downstream success in Rouge-L relative to the 1M-DOC model plotted against the average frequency in each example. Trend lines were plotted based on the entire data, but for visibility reasons the scatter is based on averages over bins containing each 2% of data.

1912; Zipf, 1949), the number of frequent words over which the tokenizers agree is quite small, In terms of types, but they cover a large portion of the 3 million document sample over which the statistics were calculated. The English words that appear at least 10^6 times in the sampled corpus, 162 in number, cover 47% of the words in the corpus. On the other hand, in Turkish, due to the thicker tail of the Zipf's distribution of morphologically rich languages, only 71 words answer this criterion, covering 26% of the corpus.

We conclude that the discrepancy in compression ability, as evident in Table 1 stems mostly from the difference in the compression in less common words. This tail of less frequent words is consisted of the semantically interesting words, so it is likely that this gap in compression causes the gaps in model performance.

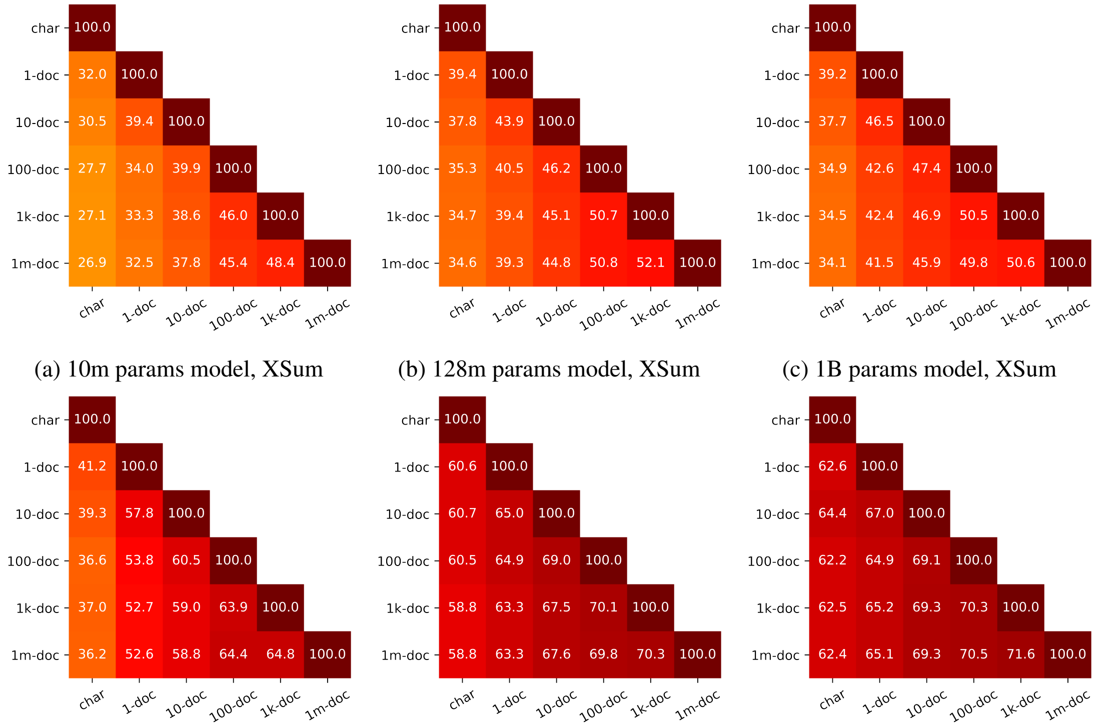
Performance over Frequent Words To complement the analysis above we also broke down the results of the generation tasks by average frequency of the words in the targeted output of each example. The results, plotted in Figure 5, include the difference in Rouge-L per example from the best 1M-DOC model per task and model size. it shows that the differences between differently tokenizing models are more pronounced over examples with rarer words.

Together, this and the previous analysis shed some light on the reasons for the correlation found in our main result. We demonstrate that the differences in performance between the various models are indeed more pronounced in the presence of rarer words, which are exactly the words that the tokenizers compress differently. It is thus highly probable that word frequency is a major confounding factor the connects compression with downstream performance.

In addition, this analysis may point to the benefit of challenge sets, comprised of examples with rarer words, in the evaluation of tokenization.

Similarity between Tokenizers The results so far compared the output of each model to the target outputs, where we showed that models are performing better when equipped with better compressing tokenizers. In order to show that the models are also converging towards similar generations, we plotted, in Figure 6, the pair-wise overlap between the outputs of all models for the English generation tasks, measured in Rouge-L .

The analysis shows that, for all tasks and model sizes, models with similarly supported tokenizers tend to output similar predictions, regardless of whether the predictions are similar to the gold targets. It is also noticeable that, in accordance with our main results, the differences in the high-support region are less pronounced than those between the less supported tokenizers.



(a) 10m params model, XSum

(b) 128m params model, XSum

(c) 1B params model, XSum

(d) 10m params model, QG-QA

(e) 128m params model, QG-QA

(f) 1B params model, QG-QA

Figure 6: Pair-wise rouge scores between outputs of the different models. Darker is higher Rouge-L scores and higher similarity between outputs. Models with similar number of supporting documents tend to output similar predictions.

7 Conclusions

In this paper we demonstrated the importance of compression to tokenization as an intrinsic evaluation of tokenization quality that indicates the performance on extrinsic downstream tasks. We argued in favor of compression-driven tokenization from a theoretical perspective, since it may make the tokenizer act as a simple standalone language model, and we showed its correlation with downstream model success.

Our experiments point to generation tasks as better downstream evaluators of tokenization since their results are both more sensitive to the tokenizer and better correlate with tokenization quality as expressed in compression ability.

In terms of linguistic diversity, the similarity in the results and analyses across two very different languages, English and Turkish, points to our conclusions being independent of specific typological characteristics. Yet, ample room is left for studying the effects of tokenization on more languages that are even more typologically diverse. Moreover, other intrinsic evaluators are still to be assessed even for the languages we did work with.

We conclude that tokenization matters, as poorly compressing tokenizers hinder the results of language models, and that investment in better compressing tokenizer has a potential of improving model performance while being relatively cheap in terms of compute. We therefore call for research to better understand the factors controlling the quality of the tokenization and its relation to overall success of the LLMs.

8 Limitations

The main limitation of this paper has to do with the amount of resources allocated to this research. Pretraining our LLMs, especially the 1B parameter models, requires a lot of compute and repeating these experiments, in a slightly different setting or just in order to replicate their results, is an expensive process.

Another limitation has to do with the limited experiments on non-English languages. Although we executed several experiments on Turkish, the cost of pretraining models of up to 1B parameters prevented us from equating the treatment given to the two languages as well as adding experiments in other non-English languages. It is possible, even if somewhat unlikely, that running the full suite of experiments on Turkish would have resulted in different conclusions. A more reasonable possibility is that running experiments on more typologically diverse languages would yield different conclusions for these languages. We mitigated this risk by choosing a language that is extremely different from English.

Acknowledgements

We thank Alon Jacovi and Uri Shaham for the helpful discussions and

References

Duygu Ataman and Marcello Federico. 2018. An evaluation of two vocabulary reduction methods for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 97–110.

Lisa Beinborn and Yuval Pinter. 2023. Analyzing cognitive plausibility of subword tokenization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4478–4486, Singapore. Association for Computational Linguistics.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

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.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics.

J-B Estoup. 1912. Gammes sténographiques. recueil de textes choisis pour l'acquisition méthodique de la vitesse, précédé d'une introduction par j.-b. estoup.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

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.

Team Gemini. 2023. Gemini: A family of highly capable multimodal models.

Edward Gow-Smith, Harish Tayyar Madabushi, Carolina Scarton, and Aline Villavicencio. 2022. Improving tokenisation by alternative treatment of spaces. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11430–11443, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. 2024. Olmo: Accelerating the science of language models.

Eylon Gueta, Avi Shmidman, Shaltiel Shmidman, Cheyn Shmuel Shmidman, Joshua Guedalia, Moshe Koppel, Dan Bareket, Amit Seker, and Reut Tsarfaty. 2023. Large pre-trained models with extra-large vocabularies: A contrastive analysis of hebrew bert models and a new one to outperform them all.

Ximena Gutierrez-Vasques, Christian Bentz, and Tanja Samardžić. 2023. Languages through the looking glass of bpe compression. *Computational Linguistics*, 49(4):943–1001.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Derivational morphology improves BERT's interpretation of complex words. 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 3594–3608, Online. Association for Computational Linguistics.

Valentin Hofmann, Hinrich Schuetze, and Janet Pierrehumbert. 2022. An embarrassingly simple method to mitigate undesirable properties of pretrained language model tokenizers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–393, Dublin, Ireland. Association for Computational Linguistics.

Omri Keren, Tal Avinari, Reut Tsarfaty, and Omer Levy. 2022. Breaking character: Are subwords good enough for mrls after all?

Stav Klein and Reut Tsarfaty. 2020. Getting the ##life out of living: How adequate are word-pieces for modelling complex morphology? In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 204–209, Online. 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.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, and Stefan Kombrink. 2012. Subword language modeling with neural networks.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. 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.

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. 2022. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*.

Tomer Ronen, Omer Levy, and Avram Golbert. 2023. Vision transformers with mixed-resolution tokenization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4612–4621.

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. 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 3118–3135, Online. Association for Computational Linguistics.

Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. 2021. Tokenlearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems*, volume 34, pages 12786–12797. Curran Associates, Inc.

Jonne Saleva and Constantine Lignos. 2023. What changes when you randomly choose BPE merge operations? not much. In *The Fourth Workshop on Insights from Negative Results in NLP*, pages 59–66, Dubrovnik, Croatia. 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.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

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.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

David Vilar and Marcello Federico. 2021. A statistical extension of byte-pair encoding. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 263–275, Bangkok, Thailand (online). Association for Computational Linguistics.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Terry Winograd. 1971. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC.

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.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Shaked Yehezkel and Yuval Pinter. 2023. Incorporating context into subword vocabularies. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 623–635, Dubrovnik, Croatia. Association for Computational Linguistics.

George Kingsley Zipf. 1949. *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books.

A Training Details

Tokenizers were trained on the first documents in C4, for English, and mC4, for Turkish, as they are ordered in the Tensorflow Datasets repository. Therefore, the data for the less-supported tokenizers is contained in the data for the better supported ones.

In all cases we limited the vocabulary size to $32k$, but in practice, for tokenizers supported by little data, the vocabulary size was lower than $32k$, since the data did not contain a sufficient number of words and subwords. Specifically, for English, the vocabulary size of 100-DOC is $23k$, of 10-DOC – $3.7k$, and of 1-DOC – $1k$. For Turkish the size of 10-DOC is $9.5k$, and of 1-DOC – $3.9k$.

In the case of CHAR, supported by no documents at all, the tokenizer simply breaks all the words into characters and replaces any foreign character with the *unknown* sign.

Models were trained using the T5X framework (Roberts et al., 2022) on the span corruption task (Raffel et al., 2020) for $200k$ training steps, with a batch size of 512. Every training example was truncated to the maximal length of 1024 tokens.[5]

[5] The fixed example length in terms of tokens leads of course to differences in the amount of data seen by the models

The models were finetuned for $4k$ steps and $20k$ steps on the classification and generation tasks, respectively, with a batch size of 128. The decoder-only models were tasks with the generation of a gold output when used for classification tasks as well. For example, in the QQP task the outputs were assumed to be either *duplicated* or *no duplicated*, where any other output considered wrong. A manual inspection showed that models learned perfectly to output one of the desired targets.

during training based on their tokenizers. We consider this as another boon of well-compressing tokenizers, since the computation budget is usually preset, as it is the case in our setting.