Wine is Not v i n.
On the Compatibility of Tokenizations Across Languages

Antonis Maronikolakis∗ and Philipp Dufter∗ and Hinrich Schütze
Center for Information and Language Processing (CIS), LMU Munich, Germany
antmarakis,philipp@cis.lmu.de

Abstract

The size of the vocabulary is a central design choice in large pretrained language models, with respect to both performance and memory requirements. Typically, subword tokenization algorithms such as byte pair encoding and WordPiece are used. In this work, we investigate the compatibility of tokenizations for multilingual static and contextualized embedding spaces and propose a measure that reflects the compatibility of tokenizations across languages. Our goal is to prevent incompatible tokenizations, e.g., ”wine” (word-level) in English vs. ”v i n” (character-level) in French, which make it hard to learn good multilingual semantic representations. We show that our compatibility measure allows the system designer to create vocabularies across languages that are compatible – a desideratum that so far has been neglected in multilingual models.

1 Introduction

Pretrained language models (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2019) have become the de-facto standard in natural language processing (NLP). Due to memory constraints and to overcome sparsity, subword tokenization models such as byte pair encoding (Sennrich et al., 2016) or WordPiece tokenization (Schuster and Nakajima, 2012) are most commonly used. Multilingual pretrained language models (PLMs) such as mBERT (Devlin et al., 2019) are mostly trained in an unsupervised fashion without any cross-lingual supervision. They rely on the fact that the underlying distributions across languages can be well aligned (Conneau et al., 2020b).

However, there is little work that investigates the compatibility of tokenizations across languages. We argue that this is an important research question for several reasons. i) Recent work suggests that the tokenization in multilingual models influences the multilinguality of these models (Rust et al., 2021). This corresponds to a strong intuition: in case of heavily diverging tokenizations, e.g., word tokenization in English and character tokenization in French it will be hard to train common cross-lingual representations. ii) Given a limited vocabulary budget, e.g., from memory constraints, a researcher is naturally interested in how to distribute the budget across multiple languages. iii) Currently, a balance of the tokenization is heuristically achieved through up- and downsampling (e.g., Conneau et al., 2020a). With a better tokenization strategy, one might be able to achieve performance improvements.

Research in multilinguality has so far focused, among other directions, on analyzing and improving cross-lingual transfer (Artetxe et al., 2020a; K et al., 2020; Wu and Dredze, 2019; Conneau et al., 2020b) and modeling tasks (Conneau and Lam-
ple, 2019; Huang et al., 2019). Earlier work on
tokenization focused on the development of algo-
rithms (Schuster and Nakajima, 2012; Gage, 1994)
while analysis of the effect of tokenization has been
undertaken both in the general (Bostrom and Dur-
rett, 2020) and in the multilingual domain (Wei
et al., 2021).

We propose to investigate the compatibility of
tokenizations across languages systematically. To
this end, we train both static and contextualized
embeddings using different vocabulary sizes and
WordPiece tokenization (Schuster and Nakajima,
2012). Subsequently we investigate the similar-
ity of the embedding spaces and their degree of
multilinguality.

Figure 1 shows an example of our research ques-
tion: here we trained static embedding spaces and
investigated the similarity of the embedding spaces
by computing the singular value gap (SVG) mea-
ure. The plots show that different vocabulary sizes
yield very different similarities. In addition, lan-
guage pairs behave quite differently. See the cap-
tion for more information.

In summary, our contributions are:¹

1. We propose an experimental setting that al-
lows the comparison of embeddings at differ-
ent tokenization levels with each other.

2. We provide evidence that vocabulary size
compatibility matters, i.e., if tokenizations
across languages are incompatible it is hard
to achieve multilingual representations.

3. We propose a measure that indicates the com-
patibility of tokenizations.

2 Related Work

In a position paper, Artetxe et al. (2020b) advo-
cated for more rigor in cross-lingual research. The
topic of tokenization was raised, especially the fact
that current word and subword tokenization algo-
rithms do not adequately capture morphological
nuances and do not do justice to the great variety
of languages in the world like those that use lo-
graphic scripts. These considerations motivate
us to propose a method that makes better use of
vocabulary space, a facet of multilingual models
that is often overlooked.

2.1 Multilingual Research

Investigation of the multilinguality of BERT
has gathered momentum in recent years. Wu
and Dredze (2019); Nooralahzadeh et al. (2020);
Artetxe et al. (2020a); Conneau et al. (2020b) in-
vestigate cross-lingual transfer, with an extensive
study of multilinguality at-scale presented in Con-
neau et al. (2020a). K et al. (2020) research how ling-
guistic variation in language properties affects mul-
tilingual BERT models, with Dufter and Schütze
(2020) further investigating the effect of model
architecture and linguistic properties on BERT’s
multilinguality. Conneau and Lample (2019) focus
on variations of modeling tasks and their effect on
downstream tasks such as XNLI (Conneau et al.,
2018) and machine translation. A novel set of cross-
lingual pre-training tasks was introduced in Huang
et al. (2019). Anastasopoulos and Neubig (2020)
show that performance in bilingual experiments is
not optimal when English is used as the “hub”, but
it depends on the language pair.

Work on the representation space of BERT has
also been abundant. Singh et al. (2019) investigate
the representation space of mBERT and its prop-
erties and Pires et al. (2019) initially researched the
degree of multilinguality of mBERT.

In our work, we further investigate the zero-
shot capabilities of bilingual models and present
a method for comparing multilingual embeddings
across tokenizations.

2.2 Tokenization Research

Another vein of research into model analysis has
been the study of the effect of tokenization. Heinz-
lering and Strube (2018) compute BPE embeddings
for 275 languages and perform a cross-lingual
study on various tokenization methods. Further
study into BPEs is conducted by Wei et al. (2021),
where byte-level BPEs are compared against their
character-level counterparts as it pertains to multi-
tilingual models while BPE vocabulary size is exam-
ined in the context of neural machine translation
by Gowda and May (2020). Bostrom and Durrett
(2020) show that byte-pair encoding is suboptimal
when pretraining language models. In our work, we
investigate another prevalent tokenization method,
WordPiece, and show that vocabulary size plays
an important role in achieving multilinguality in
BERT.

Research into the selection of tokenization al-
gorithms has produced alternative methods. For

¹Code available at https://github.com/
antmarakis/vocab_size_compat
example, Kudo (2018) present a subword regularization method where a model is trained with multiple tokenizations in order to reinforce robustness in the model; and Aguilar et al. (2020) bridge the gap between subword and character-level models, proposing a module that learns to approximate subword embeddings given characters. Asgari et al. (2020); Provilkov et al. (2020) further research subword tokenization methods. With our work we introduce a measure that can help NLP researchers pick vocabulary sizes with a more robust procedure.

3 Vocabulary Sizes

3.1 Tokenization

Let $C$ be the set of unicode code points. We define a tokenizer as a function $\tau: C^t \rightarrow V^s_\tau$ that maps a sequence of $t$ unicode code points to a sequence of $s$ vocabulary tokens. $V_\tau$ is the vocabulary of the tokenizer and $n_\tau := |V_\tau|$ is the vocabulary size of the tokenizer.

Usually the tokenizer is trained on a corpus $U$. Popular tokenizers are byte-pair encoding (Sennrich et al., 2016), WordPiece (Schuster and Nakajima, 2012) and SentencePiece (Kudo and Richardson, 2018). An example tokenization using the WordPiece tokenizer of the PLM bert-base-multilingual-cased for the input $c = \text{“Exceptional weather.”}$ is $\tau(c) = [\text{“Ex”}, \text{“##ception”}, \text{“##al”}, \text{“weather”}, \text{“.”}]$, where “##” is a continuation symbol.

Throughout this paper we use the WordPiece tokenizer. The main motivation is that important successful PLMs use this tokenization method. In future work we plan to extend this analysis to other tokenization algorithms as well.

3.2 Compression Rate

Rust et al. (2021) use two measures to compare tokenizations across languages. However, they assume that each language has the notion of “words” that are separated by whitespace. For example they compute the proportion of words that are split into multiple tokens by the tokenizer.

We posit the following desiderata for a measure that compares tokenizations across languages: i) Assume that a text consists of a sequence of un- code code points. Do not require the existence of “words”. ii) Take into account vocabulary size and alphabet size. iii) Take into account the frequency of tokens in a corpus.

The first two desiderata should ensure that the measure is as inclusive as possible with regard to the variety of languages; the last desideratum ensures that the measure considers how “important” different tokens are in a corpus. Based on these desiderata, we propose two measures, an absolute and a relative compression rate.

ACR $r^{ABS}$. Let $c \in C^t$ be a corpus with $t$ unicode code points and $\tau_n(c)$ be the tokenized version of the corpus for a tokenizer with vocabulary size $n$. The absolute compression rate is then simply

$$r^{ABS}(n) = \frac{|\tau_n(c)|}{|\tau_{n_{\min}}(c)|},$$

where $\tau_{n_{\min}}$ is the tokenizer with minimal vocabulary size $n_{\min}$. Note that the tokenization provided by $\tau_{n_{\min}}$ can be different from character tokenization and highly depends on the behavior of the tokenizer. For example for the WordPiece implementation that we use\(^2\) we obtain all unicode code

\(^2\)We use https://github.com/huggingface/tokenizers
points twice in the vocabulary, once with a continuation symbol and once without.

**RFR** $r_{REL}$: One potential disadvantage of ACR is that it does not take into consideration how many different unicode code points are used. Thus we introduce the relative compression rate that considers in addition how “inflated” the vocabulary is.

$$r_{REL}(n) = \frac{\tau_n(c)}{\tau_{n_{min}}(c)} \frac{n_{min}}{n}.$$  

One can also interpret the measure as a relative inverse type-token ratio

$$r_{REL}(n) = \frac{n}{\tau_n(c)} \frac{\tau_{n_{min}}(c)}{n_{min}}.$$  

The measures have convenient properties. For example they are monotonically decreasing and $r_{ABS}(n_{min}) = r_{REL}(n_{min}) = 1.0$. In the WordPiece implementation that we use the maximum tokenization length is limited by the number of whitespaces in the text. Thus $r_{ABS}$ is always above a language specific threshold $a$ and $r_{ABS}(n_{whitespace}) = \tau_{n_{whitespace}}(c)/\tau_{n_{min}}(c) = a$. For other tokenizers that treat whitespace as a normal unicode code point the whole text might be considered a single token, provided that the vocabulary size is sufficiently large, and thus $r_{ABS}(n_{max}) = 1/\tau_{n_{min}}(c) = a$. For the relative compression rate one can see that $r_{REL}(\infty) = 0.0$.  

Figure 2 shows both compression rates for four languages as computed on the PBC.

### 3.3 Vocabulary Size Selection

In our study we want to compare different vocabulary sizes. However, it is not trivial to come up with meaningful vocabulary sizes. Uniformly chosen sizes are not interesting as increasing the vocabulary size from, e.g., 15,000 to 16,000 in English has only a marginal effect on the actual tokenization as it mostly affects rare words. Similarly, a vocabulary size of 1000 can be decent in English, but is most likely way too small for Chinese.

We thus try to model the function $r_{ABS}$ for each language in order to be able to assess how important different vocabulary sizes are. To this end, we sample $k$ vocabulary sizes uniformly from $[n_{min}, 100000]$ and obtain data pairs $((n_1, r_{ABS}(n_1)), \ldots, (n_k, r_{ABS}(n_k)))$. We observe that the compression rate follows an exponential pattern and thus we assume the following statistical model

$$y := r_\theta(x) = \alpha(x/n_{min})^\beta + \gamma,$$

where $\theta = (\alpha, \beta, \gamma)$. Based on the asymptotic behavior of our function we assume $\gamma = a \alpha = 1 - a$. This gives us $r_\theta(x) = (1 - a)(x/n_{min})^\beta + a$ and the only parameter is $\beta$.

We fit $\beta$ with a simple linear regression without intercept, i.e., log $((y-a)/1-a) = \beta \log(x/n_{min})$. Overall we obtain an estimate $\hat{\beta}$ and thus a model $r_{\hat{\beta}}^{ABS}$.

We can now sample interesting vocabulary sizes e.g., by sampling in areas where the rate of change $(r_{\hat{\beta}}^{ABS})'$ is high. Alternatively, we go uniformly through possible compression rates: starting from a maximum compression rate we select values at 0.1-mark intervals $(1.0, 0.9, \ldots)$ until we reach the minimum compression rate. Using $(r_{\hat{\beta}}^{ABS})^{-1}$ we obtain vocabulary sizes from the chosen compression rates.

Table 1 shows selected vocabulary sizes for each language. Subscript shows ACR.

<table>
<thead>
<tr>
<th>Language</th>
<th>1291.0</th>
<th>1431.0</th>
<th>1571.0</th>
<th>6189.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>1470.9</td>
<td>1670.9</td>
<td>1910.9</td>
<td>7054.9</td>
</tr>
<tr>
<td>GR</td>
<td>1720.8</td>
<td>2010.8</td>
<td>1900.8</td>
<td>8733.8</td>
</tr>
<tr>
<td>ES</td>
<td>2090.7</td>
<td>2540.7</td>
<td>2440.7</td>
<td>1708.7</td>
</tr>
<tr>
<td>ES</td>
<td>2670.6</td>
<td>3300.6</td>
<td>3230.6</td>
<td>3378.6</td>
</tr>
<tr>
<td>ES</td>
<td>3740.5</td>
<td>4700.5</td>
<td>4710.5</td>
<td>-</td>
</tr>
<tr>
<td>GR</td>
<td>6530.4</td>
<td>8210.4</td>
<td>8270.4</td>
<td>-</td>
</tr>
<tr>
<td>GR</td>
<td>4130.3</td>
<td>29240.3</td>
<td>26420.3</td>
<td>-</td>
</tr>
<tr>
<td>GR</td>
<td>188630.29</td>
<td>510170.24</td>
<td>503410.24</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Selected vocabulary sizes for each language.

### 4 Experiments

#### 4.1 Data

The Parallel Bible Corpus (PBC) by Mayer and Cysouw (2014) is a multi-parallel corpus spanning 1259 languages and up to 30k verses per translation. We chose this corpus for three reasons: i) For a clean experimental setup in our sentence retrieval task we require a multi-parallel corpus with additional paraphrases within the same language. This is possible in PBC as there are multiple translations is possible in PBC as there are multiple translations. Dufter and Schütze (2020) provided evidence that experimenting on a small setup and verifying key results in a larger setup is feasible.  

ii) The corpus has a vast language coverage.

3Language codes follow the ISO 639-3 standard: https://iso639-3.sil.org/
We compute some general statistics (see Appendix Table 3) on PBC, calculating the maximum, average and minimum verse character length for each examined language, as well as the length covering 95% of all instances. This information was used in conjunction with computational considerations to set the maximum sequence length for our experiments.

### 4.1.1 Language Selection

We perform our static embedding analysis on all 61 complete “newworld” editions in the PBC. These are very literal translations spanning the same set of verses across languages and thus ensure high comparability. For contextualized representations, experiments are more expensive. Thus we chose to focus on languages ENG, ELL, RUS and ZHO, which span different scripts, degrees of morphology and language families.

### 4.1.2 Vocabulary Size Selection

Instead of hand-picking arbitrary sizes without guidance behind the selection process we follow our proposal from §3.3 and select sizes based on compression rates. Starting from a maximum compression rate (i.e., smallest vocabulary size) we choose sizes at 0.1-mark intervals (1.0, 0.9, . . . ) until we reach a minimum compression rate (i.e., maximum vocabulary size).

### 4.1.3 Fake-English

We also experiment using Fake-English, denoting the English to Fake-English pair by EngFake. Fake-English is a concept examined in K et al. (2020), where English characters are converted to some separate, special characters such that the model processes English and Fake-English as two different languages. The “new” data that is created exhibits the same grammar and structure as English, but the script is different. This offers both a simple upper performance limit and a scrying glass into potential bugs or shortcomings of our approach. Note that to generate the Fake-English data, we used the same edition as the English Bible set, to ensure better comparability.

### 4.2 Static Embeddings

#### 4.2.1 Training

We evaluate vocabulary compatibility in static embedding spaces. To this end we tokenize comparable corpora using learned WordPiece vocabularies with different sizes. Subsequently we obtain static embeddings using fastText (Bojanowski et al., 2017) with default parameters.

### 4.2.2 Compatibility Evaluation

Comparing static embedding spaces at different granularities is challenging. Tasks like word translation or sentence retrieval disqualify, as obtaining word or sentence representations through mean pooling is too crude to get discriminative evaluation results. Training a neural network on top of the embeddings to evaluate multilinguality requires additional input, might obliterate the effect we want to investigate (i.e., the compatibility of the embedding spaces) and is not feasible when processing many language pairs. Therefore we employ a popular method to predict the cross-lingual performance using the spectral similarity of embedding spaces. Dubossarsky et al. (2020) introduced the measure singular value gap (SVG) and showed that it correlates well with cross-lingual downstream performance such as bilingual lexicon induction. Given two embedding spaces for two languages \( X_l \in \mathbb{R}^{n_l \times d} \) and \( X_e \in \mathbb{R}^{n_e \times d} \) it is computed as

\[
SVG(X_l, X_e) := \sum_{i=1}^{d} \left( \log(\sigma^e_i) - \log(\sigma^f_i) \right)^2
\]

with \((\sigma^k_i)\) being the sorted singular values of \(X_k\). Instead of computing the sum across all dimensions we follow (Dubossarsky et al., 2020) and consider only the first 40 singular values. We found that experiments with the two alternative measures cond-hm and econd-hm that Dubossarsky et al. (2020) suggest yield similar results and thus focus only on SVG for the sake of clarity.

### 4.3 Contextualized Embeddings

#### 4.3.1 Pretraining

**Model Details.** For our experiments, we made use of BERT (Devlin et al., 2019). We performed two separate sets of experiments, one on PBC and another on the Wikipedia corpus. The Wikipedia experiments serve as a verification of the generalization of our findings to a larger scale.

A thorough and rigorous analysis on the effect of vocabulary sizes can be prohibitively costly on the larger scale. When comparing the vocabulary sizes of two languages with \(m\) and \(n\) different sizes each, we need \(m \times n\) different models. For example, to compare English \((m = 9)\) with Greek \((n = 9)\) we need to train \(9 \times 9 = 81\) models. Pretraining all
these models on Wikipedia with the default BERT parameters is prohibitively expensive.

Thus, we split our experiment into the small-scale Bible setup and the large-scale Wikipedia setup (§6). We also downsized the models for the Bible experiments. These models use a single attention head, which has been shown to have competitive performance compared to the multi-head approach with a substantial drop in required computational resources (Michel et al., 2019). We also downsized the hidden layer size, from 3072 to 256. We call this model BERT\textsubscript{bible}.

For the Wikipedia experiments, the default BERT hyperparameters were kept. We use BERT\textsubscript{wiki} to denote this model architecture.

\textbf{Bible Experiments.} For our work, we pretrained the downsized BERT\textsubscript{bible} models on the PBC data. Then we evaluated on a bidirectional sentence retrieval task after finetuning following the SBERT example (Reimers and Gurevych, 2019).

We select the English (ENG), Greek (ELL), Russian (RUS) and Chinese (ZHO) versions of the Bible. Development and test sets were defined using common verses across the languages. The training set varies slightly for each language, since we included all verses not in the dev/test sets. Details on these splits can be found in Table 5.

For pretraining, we pair ENG with ELL/RUS/ZHO. We denote these pairs with EngEll, EngRus and EngZho. For each pair, we take the Cartesian product of the corresponding vocabulary sizes. The training sets for the two languages are then concatenated and two tokenizers are trained with the examined vocabulary sizes. Model pretraining then takes place for 75 epochs using BERT\textsubscript{bible}. Batch size is set to 128 and the maximum sequence length is also set to 128. We used ADAM (Kingma and Ba, 2015), with a learning rate of 2e-3, a weight decay of 0.01 and an epsilon value of 1e-6. Three runs were performed and the results averaged.

\textbf{4.3.2 SBERT Finetuning}

Each pretrained model is finetuned under the SBERT paradigm (Reimers and Gurevych, 2019). The model learns to identify whether two sentences are similar or not. Sentence representations are computed as usual and then pushed through a pooling layer. These representations are compared with cosine similarity, with the mean-squared-error loss used as the objective function.

The main motivation for this finetuning is to obtain meaningful sentence representations from different tokenization granularities. Overall, this allows us to analyze the multilinguality of the representations even though the tokens in both languages might be very different, such as words and characters. During finetuning, the model also learns how to separate dissimilar sentences and bring similar ones closer together in the representation space, which allows us to evaluate multilinguality using cross-lingual sentence retrieval.

The data used for this task comes from the PBC development sets. For each language we use two different editions, that is, two separate translations in the same language, i.e., paraphrases. Across the two editions, the verses with the same identifier are paired and labeled as similar, while for the dissimilar examples we pair each verse from one edition to a different, random verse in the other edition. We keep this pairing within each language, to ensure that our cross-lingual evaluation setup is zero-shot.

\textbf{4.3.3 Compatibility Evaluation: Sentence Retrieval}

To evaluate these pretrained and finetuned models, we use a cross-lingual sentence retrieval task. Data for this evaluation task comes from the development sets for English and the paired language. Given an English verse, we are tasked with retrieving the corresponding verse in the paired language. A mean-pooling layer takes as input the token representations of a sentence after a forward pass and outputs the final verse representation. Then, for each English sentence, we retrieve the 10 most similar sentences from the paired language, as calculated via cosine similarity. Our evaluation measure is precision@10. This method was repeated analogously in the other direction: for each sentence in the paired language, we retrieve the most similar sentences in English. Scores from both directions are averaged and results are analyzed in §5.2.2.

Note that the cross-lingual sentence retrieval is zero-shot in the sense that any multilinguality can only stem from the pretraining as no cross-lingual information was used during SBERT finetuning.

\textbf{5 Results}

\textbf{5.1 Static Embedding Learning}

We show results for static embeddings in Figure 3. For better visualization we interpolated the results.
using LinearTriInterpolator in the Matplotlib library (Hunter, 2007). For alphabetic languages it is favorable to have comparable vocabulary sizes, which can be seen from the clear diagonal pattern of ELL, RUS and SPA. Chinese, being based on logograms, requires large vocabulary sizes in English. This makes intuitive sense: with small vocabulary sizes in English one arrives at a character tokenization and it is hard to imagine the equivalent of a Latin character or character n-gram in Chinese. Japanese exhibits a unique pattern, which can be explained by the fact that Japanese uses both syllables and logograms. Plots for all languages can be found in Appendix E.

Figure 3: Figures showing the interpolated SVG of static embedding spaces for different language pairs computed on the PBC. Blue indicates high compatibility (i.e., low SVG) and red the contrary. Black stars indicate vocabulary sizes where \( r^{ABS} \) of both languages are equal. One can see different patterns: alphabetic languages tend to favor the diagonal. Logograms require much larger vocabulary sizes in English. Japanese uses a mixture of logograms and a syllabic writing system and thus exhibits a special pattern.

Figure 4: Results showing the interpolated SVG of embedding spaces for different language pairs computed on Wikipedia.

5.2 Contextualized Embedding Learning

5.2.1 Fake-English Experiments

We begin our evaluation with the Fake-English experiments. As described in §4.1.3, we generate a modified “fake” English dataset from the English Bible corpus. These experiments act as our guide and debugging assistant. The behavior of these models can tell us a great deal about what to expect in the other experiments. Based on these results, hyperparameters were tuned both for pretraining and for finetuning. Results (on the test set) are shown in the rightmost plot of Figure 5. In the EngFake experiments, the diagonal is very prominent. Overall, performance in these experiments is quite high. On the diagonal, the precision@10 score is the highest in the bottom left corner, while it diminishes slightly towards the top right (detailed results can be found in Table 7 of the appendix). The fact that BERT is able to produce compatible representations between English and Fake-English at different tokenization granularities is in line with findings in (K et al., 2020).

5.2.2 Bible Experiments

As previously described in §4.3, we conducted experiments on PBC for the EngEll, EngRus and EngZho language pairs. Test results are shown in Figure 5 (with detailed tables in Appendix C). EngEll and EngRus both show stronger performance along the diagonal. Performance is low with small sizes (both for English and the paired
language). *EngEll* shows more consistently high precision across the diagonal, from small to large sizes. On the other hand, *EngRus* performance, while still better along the diagonal, is higher in the top right of the figure where both English and Russian vocabulary sizes are larger.

Despite these similar trends, performance in the *EngRus* experiment is higher than in *EngEll* (as indicated by the deep blue in the top right of the figure). Further, in *EngEll*, performance seems to maximize towards the top-middle section of Figure 5. For *EngZho*, instead of a diagonal we see a left-to-right pattern. The larger the English vocabulary size, the better performance is. With the smaller Chinese vocabulary sizes (bottom right of the figure), we see more consistently high results as well. Overall the results show similar trends to findings from the static embeddings in §5.1.

These results indicate language-specific patterns exist across tokenization sizes, something that can aid us in creating more compatible vocabularies.

### 6 Wikipedia Experiments

#### 6.1 Data and Task

To verify whether our findings generalize to a large-scale setting, we also evaluate on XNLI (Conneau et al., 2018). The models we have previously pretrained on the Bible corpus do not have the capacity to tackle this larger task, therefore we opted to instead pretrain new models on Wikipedia data. To this end, English, Greek, Russian and Chinese Wikipedia dumps were used. Due to a lack of computational resources, we kept only 7.4GB of English data, 3.3GB of Chinese and 6.3GB of Russian. For the Greek experiments, all articles were kept resulting in around 0.8GB of data in total. For all other languages, articles were randomly sampled. Details on data size can be found in Table 6 of the appendix.

Statistics for XNLI are shown in Table 4 of the appendix. Large differences were observed between the two text fields (premise and hypothesis), as well as between Chinese and the other languages (attributed to the difference in scripts).

#### 6.2 Training

To accommodate the larger dataset, BERT\textsubscript{wiki} was used. The number of epochs was set to 1 and the maximum sequence length was reduced to 128 due to computational restrictions. Analogously to the small-scale Bible experiments, for pretraining, English data was concatenated with the paired language data, with separate tokenizers trained on the two corpora.

Because evaluating for all possible combinations of size pairs is prohibitively resource-consuming, we settled on three size pairs for each language. These pairs were selected according to the Bible evaluation we ran beforehand. Namely, for each language, we picked the best, the one at the 75th percentile (perc\textsubscript{75}) and the worst pair sizes to compare according to their sentence retrieval precision@10 score.

After the pretraining of these models on Wikipedia, we finetuned them on the English XNLI training set and then performed cross-lingual zero-shot evaluation on the other languages. As before, Greek, Russian and Chinese were evaluated.

#### 6.3 Evaluation and Results

Figure 4 shows the same experiments as described in §5.1, this time conducted on the Wikipedia dataset. The key trends are observable here as well.

In Table 2 we show the accuracy of the three examined pairs (as defined in §6.2: best, perc\textsubscript{75}, worst) for each language in the XNLI task. Note that a majority baseline has an accuracy of 1/3. The pairs are sorted according to their performance in the Bible experiments (precision@10 on sentence
Table 2: Zero-shot XNLI accuracy for English (source) and three target languages. We give the vocabulary sizes and, as a subscript, Absolute Compression Rate (ACR). In each block of three results (one each for ELL, RUS, ZHO), the pair of vocabulary sizes with the best performance on sentence retrieval is at the top, followed by perc_75 and the worst performing pair. Wikipedia results are a good predictor of XNLI performance for Greek and Russian. See text for more details.

<table>
<thead>
<tr>
<th>Language Pair</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EngEll</td>
<td></td>
</tr>
<tr>
<td>14098_0.4 - 19789_0.3</td>
<td>48.8</td>
</tr>
<tr>
<td>7414_0.9 - 12539_0.4</td>
<td>46.4</td>
</tr>
<tr>
<td>22158_0.3 - 6177_0.9</td>
<td>44.6</td>
</tr>
<tr>
<td>EngRus</td>
<td></td>
</tr>
<tr>
<td>22158_0.3 - 15892_0.3</td>
<td>59.1</td>
</tr>
<tr>
<td>9752_0.6 - 15892_0.3</td>
<td>58.3</td>
</tr>
<tr>
<td>69561_0.1 - 15892_0.3</td>
<td>56.4</td>
</tr>
<tr>
<td>EngZho</td>
<td></td>
</tr>
<tr>
<td>22158_0.3 - 43005_0.9</td>
<td>45.1</td>
</tr>
<tr>
<td>14098_0.4 - 39393_1.0</td>
<td>44.7</td>
</tr>
<tr>
<td>69561_0.0 - 81426_0.7</td>
<td>44.8</td>
</tr>
</tbody>
</table>

retrieval, largest at the top). For all language pairs, performance on the English XNLI test set was similar, slightly fluctuating (±0.75) around 74%. These scores are hence omitted from Table 2 since they do not add much to the discussion.

The best combinations (from the Bible experiments) for EngEll and EngRus have the best XNLI performance as well, followed by perc_75 and finally the worst-performing combination. Thus, our findings for EngEll and EngRus are corroborated: the best performances are found when the absolute compression rate is similar.

For EngZho, we see little variation between the three chosen size combinations although the best-performing combination is still at the top. Maybe this is due to the fact that Chinese behaves differently from the two “alphabetic” languages and therefore a more extensive hyperparameter optimization would be required. We leave this question for future research.

6.4 Correlation Analysis

One major objective is to be able to compare the compatibility of tokenizations across languages. To this end we compute $\log(r_e^{ABS})/\log(r_f^{ABS})$ for two languages $e$, $f$ and analogously for $r_e^{REL}$. We compute the correlation of this compatibility measure using the original mBERT tokenizer with the downstream performance of mBERT on XNLI as reported by (Hu et al., 2020). More details can be found in the supplementary.

The Pearson correlation is .40 and .34, respectively. This indicates that our measures capture the compatibility of tokenizations and correlate with zero-shot transfer downstream performance. The strength of the correlation is similar to what Rust et al. (2021) find.

7 Conclusion

We investigated tokenization compatibility across languages, both for static and contextualized embeddings. To this end, we proposed compression rates and a method to select meaningful vocabulary sizes in an automated manner for any language. We introduced a compatibility measure and showed that it correlates with downstream performance.

We gave evidence for two key findings that hold for both static and contextualized embeddings. i) Tokenization compatibility can have a significant impact on multilingual performance: performance is generally higher when vocabulary sizes are compatible. ii) Tokenization compatibility varies significantly across language pairs: pairs of alphabetic languages show a stronger performance on the diagonal (i.e., for comparable vocabulary sizes) while an alphabetic language like English requires large vocabulary sizes when paired with a logographic language like Chinese.

Our study has clear limitations: we only experimented with the WordPiece tokenizer and with a small number of language pairs and there are only a few larger-scale experiments on XNLI. We will continue research in this direction and hope that this paper will spark interest by other researchers in the compatibility of tokenizations across languages and its effect on multilinguality.

Acknowledgements

This work was supported by the European Research Council (# 740516). The second author was supported by the Bavarian research institute for digital transformation (bidt) through their fellowship program. We thank Ehsaneddin Asgari and the anonymous reviewers for valuable comments and fruitful discussions.

References

Gustavo Aguilar, Bryan McCann, Tong Niu, Nazneen Fatema Rajani, Nitish Shirish Keskar, and


A Data Statistics

Here we present details for the PBC and Wikipedia data.

<table>
<thead>
<tr>
<th>Language</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG</td>
<td>23,633</td>
<td>5,000</td>
<td>2,500</td>
</tr>
<tr>
<td>ELL</td>
<td>23,673</td>
<td>5,000</td>
<td>2,500</td>
</tr>
<tr>
<td>RUS</td>
<td>23,673</td>
<td>5,000</td>
<td>2,500</td>
</tr>
<tr>
<td>ZHO</td>
<td>23,657</td>
<td>5,000</td>
<td>2,500</td>
</tr>
</tbody>
</table>

Table 3: Character length statistics for the PBC data.

<table>
<thead>
<tr>
<th>Language</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG</td>
<td>7436MB</td>
</tr>
<tr>
<td>ELL</td>
<td>809MB</td>
</tr>
<tr>
<td>RUS</td>
<td>6291MB</td>
</tr>
<tr>
<td>ZHO</td>
<td>3252MB</td>
</tr>
</tbody>
</table>

Table 6: Sizes for the languages in our Wikipedia data.

B Computational Details

For each individual PBC experiment, pretraining took around 2 hours and finetuning/evaluation 25 minutes. For each individual Wikipedia experiment, pretraining took around 145 hours and fine-tuning/evaluation 5 hours. A multi-GPU server was used, with GeForce GTX 1080Ti devices.

C Bible Experiments Detailed Results

Here we present the detailed tables for the PBC results. Three runs were made with results averaged. Standard deviations were small, so they were omitted for readability.

Table 7: Similarity Matrix for EngFake. Rows denote English and columns Fake-English.

Table 8: Similarity Matrix for EngEll. Rows denote English and columns Greek.

Table 9: Similarity Matrix for EngRuS. Rows denote English and columns Russian.

Table 10: Similarity Matrix for EngZho. Rows denote English and columns Chinese.
D  Correlation Analysis

We show detailed results for the correlation analysis in Table 11.

E  SVG Plots

In the next pages, we show the complete SVG plots for all 61 languages.
Figure 6: Figures showing the interpolated SVG for different language pairs as computed on the PBC (1/4).
Figure 7: Figures showing the interpolated SVG for different language pairs as computed on the PBC (2/4).
Figure 8: Figures showing the interpolated SVG for different language pairs as computed on the PBC (3/4).
<table>
<thead>
<tr>
<th>Language</th>
<th>XNLI Acc.</th>
<th>$r^{ABS}$</th>
<th>$r^{REL}$</th>
<th>$\log(r_x^{ABS}) / \log(r_{eng}^{ABS})$</th>
<th>$\log(r_x^{REL}) / \log(r_{eng}^{REL})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ara</td>
<td>64.9</td>
<td>0.46</td>
<td>0.06</td>
<td>0.64</td>
<td>0.68</td>
</tr>
<tr>
<td>bul</td>
<td>68.9</td>
<td>0.39</td>
<td>0.03</td>
<td>0.77</td>
<td>0.85</td>
</tr>
<tr>
<td>deu</td>
<td>71.1</td>
<td>0.31</td>
<td>0.01</td>
<td>0.95</td>
<td>1.06</td>
</tr>
<tr>
<td>ell</td>
<td>66.4</td>
<td>0.57</td>
<td>0.06</td>
<td>0.46</td>
<td>0.65</td>
</tr>
<tr>
<td>eng</td>
<td>81.4</td>
<td>0.30</td>
<td>0.01</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>esp</td>
<td>74.3</td>
<td>0.32</td>
<td>0.02</td>
<td>0.95</td>
<td>0.93</td>
</tr>
<tr>
<td>fra</td>
<td>73.8</td>
<td>0.33</td>
<td>0.02</td>
<td>0.91</td>
<td>0.95</td>
</tr>
<tr>
<td>hin</td>
<td>60.0</td>
<td>0.53</td>
<td>0.08</td>
<td>0.53</td>
<td>0.61</td>
</tr>
<tr>
<td>rus</td>
<td>69.0</td>
<td>0.37</td>
<td>0.02</td>
<td>0.82</td>
<td>0.90</td>
</tr>
<tr>
<td>swa</td>
<td>50.4</td>
<td>0.39</td>
<td>0.03</td>
<td>0.78</td>
<td>0.84</td>
</tr>
<tr>
<td>tha</td>
<td>55.8</td>
<td>0.68</td>
<td>0.12</td>
<td>0.31</td>
<td>0.49</td>
</tr>
<tr>
<td>tur</td>
<td>61.6</td>
<td>0.37</td>
<td>0.03</td>
<td>0.81</td>
<td>0.81</td>
</tr>
<tr>
<td>urd</td>
<td>58.0</td>
<td>0.48</td>
<td>0.07</td>
<td>0.60</td>
<td>0.63</td>
</tr>
<tr>
<td>vie</td>
<td>69.5</td>
<td>0.45</td>
<td>0.08</td>
<td>0.65</td>
<td>0.60</td>
</tr>
<tr>
<td>zho</td>
<td>69.3</td>
<td>0.96</td>
<td>0.69</td>
<td>0.04</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Table 11: Compression rates computed using the mBERT tokenizer. XNLI results by Hu et al. (2020).

Figure 9: Figures showing the interpolated SVG for different language pairs as computed on the PBC (4/4).