Clustering Monolingual Vocabularies to Improve Cross-Lingual Generalization

Riccardo Bassani
Utrecht University
Netherlands
r.bassani@students.uu.nl

Anders Søgaard
University of Copenhagen
Denmark
soegaard@di.ku.dk

Tejaswini Deoskar
Utrecht University
Netherlands
t.deoskar@uu.nl

Abstract

Multilingual language models exhibit better performance for some languages than for others (Singh et al., 2019), and many languages do not seem to benefit from multilingual sharing at all, presumably as a result of poor multilingual segmentation (Pyysalo et al., 2020). This work explores the idea of learning multilingual language models based on clustering of monolingual segments. We show significant improvements over standard multilingual segmentation across nine languages on a question answering task, both in a small model regime and for a model of the size of BERT-base.

1 Introduction

Since its release in 2018, multilingual BERT (mBERT) has been extensively analyzed. For instance, mBERT has been shown to facilitate zero-shot cross-lingual transfer, but transfer performance is significantly worse for typologically distant languages (Pires et al., 2019). This is because cross-lingual generalization is only partial, with some language-specific partitioning of the representational space (Libovický et al., 2019; Singh et al., 2019; Pires et al., 2019).

Moreover, mBERT relies on a single, multilingual tokenizer to create its vocabulary (WordPiece, Schuster and Nakajima (2012)), a subword tokenizer based on an algorithm similar to BPE (Gage, 1994). The multilingual tokenizer lets subwords be shared across different languages, but the approach penalizes outlier languages, whose words end up being segmented according to statistics derived from high-resource languages (Pyysalo et al., 2020; Chung et al., 2020).

In this study, we propose a novel method for building a multilingual vocabulary that is more fair to outlier languages: The core idea is *to train mBERT-like language models on multilingual clusters of monolingual tokens derived from monolingual tokenizations into vocabularies of equal size*. We argue this approach brings us the best of two worlds: monolingual (dedicated) tokenization and multilingual input representations.

Contributions Our main contribution is devising a novel way of learning multilingual vocabularies that are more fair to outlier languages. Our approach relies on clusters, and we analyze these clusters in depth. Next, we induce a multilingual mBERT-like language model from cluster representations of input sentences and evaluate this language model on the downstream task of cross-lingual question answering. We show that the model trained on our cluster representations improves over our baseline, trained with standard mBERT tokenization, across nine typologically diverse languages. We observe the same improvements across two different language model sizes.

Related work Pyysalo et al. (2020) showed how training monolingual BERT models instead of a single multilingual one can improve performance for some languages even if the amount of training data decreases. A possible cause of this phenomenon could be identified in the better quality of tokenization that can be reached when a dedicated monolingual vocabulary is used. This thesis is also supported by the findings of Conneau et al. (2020) on the importance of vocabulary size to a multilingual language model's performance.

In order to overcome the issues deriving from the use of a single multilingual vocabulary, Chung et al. (2020) propose to cluster together similar languages and to learn vocabularies for language clusters. Their methods proves to be effective, with a significant reduction of maximum description length, and significant improvements on QA, NER and XNLI tasks. This suggests that multilingual model can benefit from the use of a vocabulary that more fairly represents all languages.

In this paper, we take this approach one step further and propose a method which clusters the

Proceedings of the 1st Workshop on Multilingual Representation Learning, pages 32–40

output of monolingual tokenizations, merging dedicated vocabularies into a single vocabulary of token clusters.

2 Multilingual Vocabulary Induction

The key idea of our method is to build a single multilingual model from the output of dedicated, monolingual tokenizers, i.e., language-specific subword vocabularies, by clustering the obtained subwords by semantic similarity. A single model is then trained on the desired number of clusters, learning an embedding for each cluster. This allows for implicitly using a very large vocabulary without increasing model complexity, hopefully leading to better segmentation for outlier languages and better generalization across languages.

Our approach involves three steps:

(i) Creation of monolingual subword vocabularies.

(ii) Clustering monolingual subwords into multilingual subword clusters.

(iii) Training a multilingual language model on multilingual subword clusters and evaluating it on a downstream task.

These steps are described in turn below. An overview of the entire process is given in Figure 2.

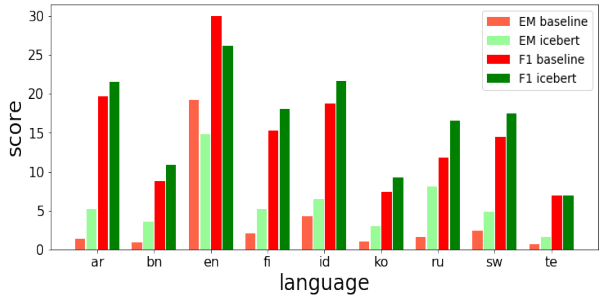


Figure 1: EM and F1 scores for the baseline and the ICEBERT model, across nine languages.

2.1 Vocabulary creation

First, monolingual vocabularies were created starting from the BPEmb vocabularies offered by Heinzerling and Strube (2018). BPEmb[1] is a collection of BPE-based subword embeddings in 275 languages. Monolingual vocabularies are available in different sizes. We extracted vocabularies of size 30k[2]

[1] //bpemb.h-its.org/

[2] This vocabulary size was chosen to be similar to the size of the monolingual BERT's vocabulary, leaving the 30k most frequent tokens in each language.

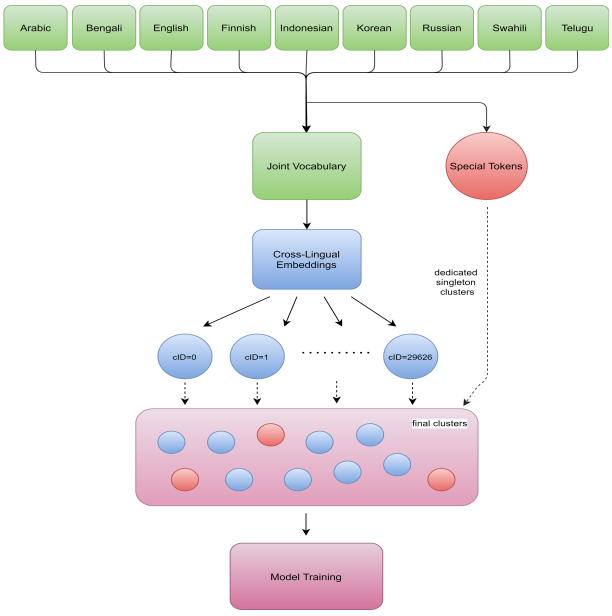


Figure 2: Overview of the three phases of the project. From top to bottom: joint vocabulary creation, subword clustering, and model training. The clusters are identified by clusters IDs (cIDs), and are represented as circles in the figure. Blue circles represent clusters obtained by partitioning the translation graph, while red circles represent singleton clusters dedicated to special tokens.

for the 9 languages present in the TyDiQA benchmark (Clark et al., 2020): Arabic, Bengali, English, Finnish, Indonesian, Korean, Russian, Swahili, and Telugu. These languages are typologically very different and as such provide for downstream evaluation of multilingual language models.

A joint vocabulary was created by concatenating the mono-lingual vocabularies and marking subwords with a language identifier. Digits, punctuation and other special symbols, which are largely language-agnostic, were not marked this way, but assigned dedicated, singleton clusters. Following Dufter and Schütze (2020), we mapped all digits to zero.

2.2 Subword clustering

The goal of subword clustering is to obtain clusters made of tokens from different languages that are semantically similar. The ideal clusters will therefore contain tokens which are near-translations of each other. Moreover, we want clusters to be generally small and of similar size. Large clusters are undesirable, since they tend to be semantically vague or inconsistent. Therefore, while allowing some variance in cluster size, we limit the maximum cluster size.

Cross-lingual embeddings creation The subwords in the joint vocabulary were clustered by semantic similarity exploiting the already available BPEmb-embeddings. We started from the monolingual BPEmb embeddings of size $d = 300$, and projected them into a shared semantic space by mapping all vectors but the English ones into the English vector space. The mapping was performed using the scripts and seed dictionaries published by Glavas et al. (2019).[3] In particular, we relied on the PROC algorithm to solve the Procrustes problem (Schönemann, 1966). Seed dictionaries were available for English-Russian and English-Finnish, containing 5000 training pairs and 2000 test pairs. Dictionaries for the remaining languages were derived from Google Translate and can be shared for resource purposes upon request.[4]

The mapping provides us with 300-dimensional multilingual embeddings, which we evaluate on a bilingual lexicon induction (BLI) task. For each source/target language pair, and for each word pair in the rtest dictionary, the top k translations of the source word are extracted from the target language vocabulary as its k nearest neighbours, measured by cosine similarity. Precision at k (p@k) scores are used to quantify the fraction of times in which the gold translation (the one in the test dictionary) is present among the top k extracted translations. The mean reciprocal rank (MRR) gives an idea of the average rank of the gold translation among the predicted translations, as expressed by the formula $\frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$, where Q is the set of all the tokens in the target vocabulary. The created multilingual embeddings achieve good average scores on BLI (p@1=0.193, p@5=0.384 p@10=0.454, MRR=0.283).

Translation graph creation Instead of simply clustering the tokens considering only the distance between embeddings, we aim to carefully create small clusters of similar sizes, ideally containing synonyms and translations.[5] To this end, we create a sparse graph containing for each subword, the top-k translations in each language, i.e., the top-k nearest neighbors in the projected space. We eliminate from this sparse translation graph all non-symmetric edges, so that the obtained graph contains an edge between subword_1 and subword_2 if and only if subword_1 is among the top-k translations of subword_2, *and* vice versa.

To select the most adequate value for k, we examine the distribution of clique sizes in each graph. k must be high enough to avoid having many subwords which are not in (non-single-ton) cliques. If k is too high, however, cliques may contain subwords that are not reciprocal translations/synonyms.

Figure 3 shows how, as expected, the number of singletons (cliques containing only one subword) decreases as k increases. We select $k = 5$, which is the smallest value that gives us a reasonably low number of isolated subwords: With $k = 5$, there are 1232 isolated subwords, requiring 1232 dedicated clusters.

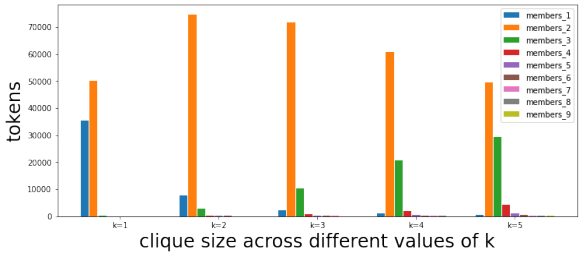
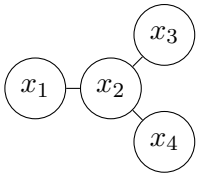


Figure 3: Number of tokens belonging to a clique of size at most s, for s in [1,9] and for different values of k. The blue bar represents tokens belonging only to singletons, the orange bar tokens belonging to cliques of size at most 2, the green one tokens belonging to cliques of size at most 3, etc.

Note that cliques do not capture all translations, as shown in the example below, where x_2 has three translations despite not being in any clique of size higher than 2.



In Figure 4, we plot the average number of languages per clique, across different clique sizes. We see that most small cliques contain translation pairs, not merely synonym pairs in the same language.

Graph Partitioning In order to obtain final clusters of similar size, we apply the METIS algorithm on the sparse translation graph. The METIS algorithm was first introduced in 1995 (Karypis and

[3] https://github.com/codogogo/xling-eval

[4] Glavas et al. (2019) showed that by applying a bootstrapping algorithm it is possible to obtain similar performances with seed dictionaries containing only 1000 pairs. Artetxe et al. (2018) also discuss inducing dictionary seeds in a completely unsupervised manner.

[5] We ran early experiments to verify that simple k-means clustering performs poorly.

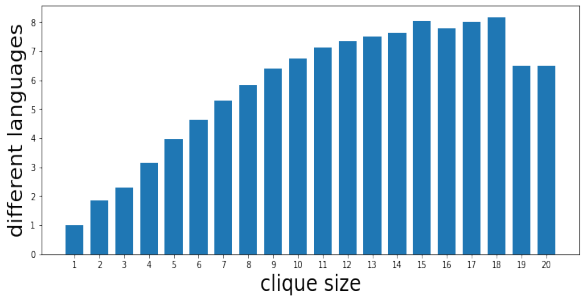


Figure 4: Average number of different languages in a clique, across different clique sizes

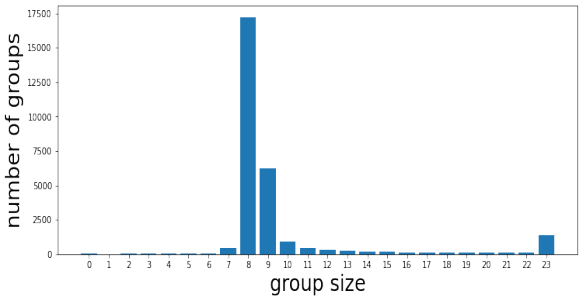


Figure 5: Distribution of the sizes of the groups in the partition obtained with the (28500, 1.5) METIS algorithm. Most groups have size 8, close to the number of languages.

Kumar, 1995), and is now available both as a standalone software[6] and as a Python package.[7] It solves the multiway graph partitioning problem, which consists in finding a partition of the vertex set into a given number of balanced sets while minimizing cut weight (Hashimoto et al., 2010).

The main parameters of the METIS balanced graph partitioning algorithm are two:

1. **p**: the number of parts in which the graph must be divided. We set this value to 28500. This was done to reach a final number of clusters of approximately 30000, considering that 665 clusters are reserved for the special tokens, and the post-processing performed on the partition.

2. **v**: the load imbalance tolerance, defining that the cluster size must not be higher than $\left\lceil \frac{N}{p} * (1 + v) \right\rceil$, where **N** is the number of nodes in the graph and **p** the number of parts the graph must be split into. This value was set to 1.5, in order to allow for a moderate degree of flexibility with the clusters size.

With these parameter values, the algorithm produced a partition made of groups of size varying between 1 and 23 (the 81 empty groups were immediately removed). Figure 5 shows that the large majority of the groups have size 7, 8 or 9, which, since we are clustering subwords from 9 languages, is exactly the desired size.

The METIS partitions contain only one group of size 1. How can it be explained, if the previous graph analysis detected the presence of 1,232 isolated subwords? The answer is that the "cut" objective of the METIS algorithm simply aims to minimize the number of edges cut by the partition. Therefore, it is not necessarily penalized when grouping together disconnected nodes. This holds not only for the METIS algorithm, but for k-way graph partitioning in general, its goal being minimizing the number of edges whose incident vertexes belong to different subsets (Karypis and Kumar, 1995). Despite this, however, graph partitioning algorithms prefer grouping together highly connected nodes, therefore the METIS algorithm can prove to be adequate for our task. The specific problem of isolated subwords can be solved by removing from the partition the 1,232 isolated subwords detected before, and assigning them to dedicated clusters. Upon doing so, we analyzed whether the so-many groups of size 7, 8, and 9 actually reflected the graph structure, or whether they were an artifact of the METIS algorithm?

To answer this question, we looked at the distribution of shortest-path lengths across all pairs of subwords belonging to the same group. Figure 6 shows that most of the subwords inhabiting the same clusters are immediate neighbours (length=1) or share a neighbour (length=2).

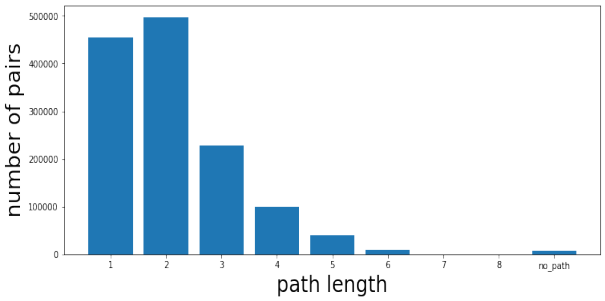


Figure 6: Distribution of shortest path lengths across intra-cluster pairs of tokens. Clusters of size 1 were not considered.

[6] http://glaros.dtc.umn.edu/gkhome/metis/metis/overview
[7] https://github.com/networkx/networkx-metis

Only around 100 intra-cluster pairs are not connected. We split the 19 groups containing such pairs into 39 groups so that all their members were connected. Finally, 665 singletons were added for the multilingual special tokens. This resulted in a total of 30292 clusters containing 270314 subwords. Some randomly selected clusters are reported in Figure 7.

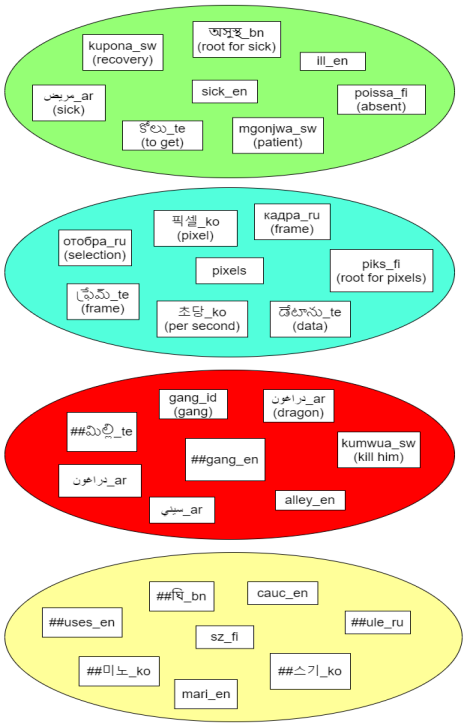


Figure 7: Examples of clusters created by the METIS algorithm. Each token is reported both in the original language and in English, except from the cases when providing an English translation is not possible. This happens for many strict subwords in the red and in the yellow clusters. For most clusters it is possible to identify an underlying concept: sick (green), pixel/data (cyan), gang/violence (red), while it is infeasible for some other clusters, especially the ones mainly made of strict subwords (yellow).

3 Model Training

Once the clustering was completed, a BERT vocabulary was built extending the set of cluster IDs (cIDs) with BERT special tokens, to give a vocabulary size of 30397, similar to the one of the original BERT model. We called our model ICEBERT, an acronym for Interlingual-Clusters Enhanced BERT.

INPUT:	Goodmorning world!
TOK.:	[‘good_en’ ‘##morning_en’ ‘world_en’ ‘!_en’]
CIDS:	28121 5132 14568 29913

Table 1: Input example tokenization and rewriting

3.1 ICEBERT training

To train our ICEBERT model on cluster IDs instead that actual subwords, we decided to take the following approach:

At training time, the entire training corpus was mapped to cluster IDs (cIDs). Each sentence was first tokenized using a lowercase BPEmb monolingual tokenizer. Before calling the tokenizer, the sentences were lowercased and digits replaced by zeros. After the tokenization was completed, all tokens were marked with a language ID. Each token was then mapped to its cID, using the token → cID dictionary created in the clustering phase. The model was then trained with a fictitious tokenizer, whose vocabulary contained the string representations of cIDs: “0", “1", ... “30291". In this way, the model’s tokenizer simply separates strings of cIDs according to white spaces, while the optimal subword tokenization is made by language-specific tokenizers. See example in Table 1.

At inference time, the same actions are performed. This means that the input to the model must be mapped to cIDs before being fed to the model. This is fairly simple and quick. For some downstream tasks, the model prediction must be modified to take into account the differences between the original input and the mapped one (e.g., span indexes in question answering). The greatest challenge would occur in generation tasks, since generated cIDs can correspond to multiple subwords. In this paper, we limit evaluation to question answering.

3.2 Baseline Training

As our baseline tokenizer, we trained a SentencePiece[8] model on a balanced Wikipedia corpus of 100k sentences, covering all the 9 languages in the experiment. Early experiments showed that uncasing the input data led to a drop in performance, as also reported in mBERT’s documentation.[9] The baseline vocabulary size is set to be the same as the ICEBERT’s one.

[8] https://github.com/google/sentencepiece

[9] https://github.com/google-research/bert/blob/master/multilingual.md

3.3 Small-Scale Pretraining

We first adopted the quick experimental setup[10] proposed by Dufter and Schütze (2020) to pre-train a small-BERT model using a training corpus of 180k sentences from Wikipedia (20k per language). The sentences were shuffled to avoid any bias towards a specific language. The model was pretrained for only 10 epochs (*vs.* 100 in Dufter and Schütze, 2020) to account for the larger amount of training data (around ten times more). All the remaining hyperparameters were kept as in the original small-BERT article except from the batch size, which was reduced to 128. The training required less than ninety minutes on a single Tesla T4 GPU. As a baseline, we pretrained a model of the same size, with the same hyperparameters, and on the same training corpus[11], omitting the mapping to cIDs.

The models were finetuned for 20 epochs, with an AdamW optimizer with learning rate equal to 5e-5. We trained both the baseline and the model five different times, and report average results for each language in Figure 8.

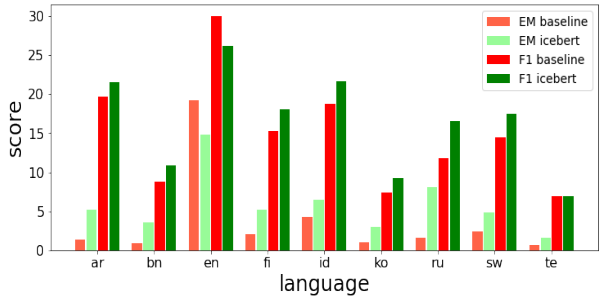


Figure 8: EM and F1 scores for the small baseline and ICEBERT model, across 9 languages.

3.4 Large-Scale Pre-training

The positive results obtained in the small-scale experiment motivated the training of a large ICEBERT model.

Configuration In the large-scale setting we trained two models of the size of the original BERT model, $\text{BERT}_{\text{BASE}}$. Our model, $\text{ICEBERT}_{\text{BASE}}$, therefore has an hidden size of 768, an intermediate size of 3072, 12 hidden layers and 12 attention heads. We mostly used the training hyperparameters reported by (Devlin et al., 2018) in the BERT paper: an AdamW optimizer (lr=1e-4 with linear decay, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay=0.01, epsilon=1e-6, warmup steps=10,000, dropout probability=0.1), 1,000,000 of training steps, a maximum sequence length of 512 and a batch size of 64 (8 per core).

Training Corpus As training corpus, we concatenated the Wikipedia corpora of all the 9 languages included in the study. To compensate for the different Wikipedia sizes across languages, we took an approach similar to the one adopted by Devlin et al. (2018). They performed exponentially smoothed weighting of the data during pretraining data creation. This means that they sampled sentences according to a multinomial distribution $q_{ii=1...N}$, where:

$$q_i = \frac{p_i^\alpha}{\sum_{j=1}^{N} p_j^\alpha} \quad \text{being} \quad p_i = \frac{n_i}{\sum_{j=1}^{N} n_j} \quad ; \tag{1}$$

The parameter $\alpha \in (0, 1)$ controls the degree to which low-resource languages are over-sampled: the closer it is to zero, the closer the number of lines across different languages. Devlin et al. (2018) chose a value of $\alpha = 0.7$ in their paper. Successive works, however, used lower values of α (Lample and Conneau, 2019, Conneau et al., 2020). Xue et al. (2020) recently showed how exponential sampling works best with a value of $\alpha = 0.3$. We therefore used $\alpha = 0.3$ to compute the values of the multinomial distribution $q_{ii=1...N}$. We then created an augmented corpus by keeping fixed the size of the English corpus and duplicating the other languages' data in order to achieve the desired ratios between corpora. During this process we also concatenated/split consecutive lines in the corpus so that each line was made of approximately 512 tokens. This was necessary since the Transformers library handling training on TPUs does not support automatic creation of balanced input segments, but requires the input dataset to be provided as a line by line dataset, i.e. so that each line corresponds to a sequence. If a line is longer than the maximum sequence length (512 in our case), it is truncated. If it is shorter, it is padded with [PAD] tokens. Wikipedia lines exhibit a great variance, ranging from single words to full paragraphs. To avoid waste of data, and that most of the input sequences were made predominantly by [PAD] tokens, we pre-processed the corpus as described above. The ICEBERT cIDs corpus was created after this pre-processing step, so that the two corpora

[10] hidden layers=12, hidden size=64, intermediate size=256, attention head=1. Total paramaters=2612989.

[11] shuffled using the same random indexes

contained exactly the same lines. The corpora were finally shuffled according to a shared indexes list. This resulted in a training corpus of approximately 80GB for the baseline and 70GB for ICEBERT[12].

Pretraining Costs It is important to notice that, despite the total number of training steps being the same as in $\text{BERT}_{\text{BASE}}$, the lower batch size implies that the model is actually trained on $\frac{1}{4}$ of the training data. A full-training was not feasible with the given resources (almost 4 weeks per model on a single v3-8 TPU), but we decided to undertrain rather than downscaling, based on the findings of Li et al. (2020), who claimed that larger models tend to achieve better performance than smaller models when trained for the same time.

The training took approximately 5 days per model on a GCP cluster made of a nd2-highmem-8 compute engine (8 vCPUs, 64 GB memory) accelerated by a v3-8 TPU node (8 cores, 128 GiB of TPU memory). The cost of the training on Google Cloud Platform amounted to little more than 60$ per model for the compute engine, while the TPUs where made available for free by Google as part of the TRC program[13]. The costs covered by this offer amount to 300$ per model if using a pre-emptible v3-8 TPUs, or to 1000$ per model if using an on-demand device.

Fine-Tuning The models were finetuned for 2 epochs, with an AdamW optimizer with a learning rate of $1e{-}5$, following Hu et al. (2020) approach[14]. We finetuned both the baseline and the model three different times.

3.5 Evaluation

We evaluate our models on TyDiQA-GoldP (Clark et al., 2020), a question answering task covering the nine (9) target languages. The model input is a question and a background passage, called *context*, which contains the answer, and the task is to predict the contiguous span of characters that human annotators have indicated, provides an answer to the question. Models are evaluated with the two metrics proposed in the SQuAD 1.1 paper (Rajpurkar et al., 2016):

1. EM: Exact match, measures the percentage of predictions exactly matching any one of the ground truth answers.
2. F1: (Macro-averaged) F1 score. Computes the F1 score between the prediction and ground truth answers, treating them as bags of tokens. For each prediction, the maximum score over all the ground truth answers is taken. The scores are then averaged over all the questions.

Our models were fine-tuned only on the English TyDiQA training dataset and applied to unseen languages at test time. This protocol is commonly referred to as unsupervised cross-lingual transfer or zero-shot cross-lingual learning.

The fine-tuning data comprises 231 batches of size 16. To train the ICEBERT model, the English dataset was mapped to cluster-IDs, following the method described in Section 3.1. The input labels, i.e., the span indexes, also had to be mapped to match the indexes of the mapped answers in the mapped contexts. The text in the test dataset was mapped to cluster IDs in the same way as for the training dataset. The span indexes output by the model were then adjusted to extract the correct text span in the original contexts, and the extracted answers were evaluated with EM and F1.

4 Results

Figures 9 shows the EM and the F1 scores across all nine languages, for the base architecture. Average results are reported in Table 2, which also gives the performances of the small models and of the fully trained mBERT model, as reported by Hu et al. (2020), for comparison.

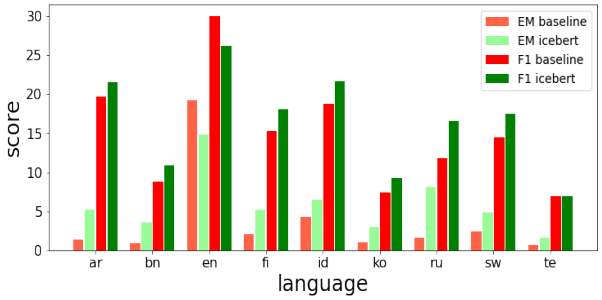


Figure 9: EM and F1 scores for the baseline and the ICEBERT model, across nine languages.

5 Discussion and Future Work

ICEBERT significantly improves over the baseline, across both the experimental settings and

[12] In UTF-8 encoding foreign characters often takes more than 8 bits, therefore by mapping foreign characters to digits the size of the corpus is reduced. The size of the English corpus, on the other hand, is increased, since many short tokens (e.g. punctuation) are mapped to 5-digits cluster IDs.

[13] https://sites.research.google/trc/

[14] The learning rate was adjusted to take into account the difference batch size (16 instead of 32).

model	EM	F1
$\text{Baseline}_{\text{SMALL}}$	1.80	12.89
$\text{ICEBERT}_{\text{SMALL}}$	**4.74**	**15.30**
$\text{Baseline}_{\text{BASE}}$	10.96	21.49
$\text{ICEBERT}_{\text{BASE}}$	**11.36**	**23.18**
mBERT	41.46	57.74

Table 2: Exact Match and F1 average scores across all languages different from English. On top of the baseline's and the ICEBERT model's performance, the scores obtained by the fully-trained mBERT model are also reported.

evaluation metrics. The base architecture F1 improvements moving from baseline tokenization to ICEBERT tokenization (1.7 F1) are encouraging. $\text{ICEBERT}_{\text{BASE}}$ improves over the baseline over almost all languages. Languages which are typologically distant from English, like Arabic, Russian, and Telugu, benefit the most from the clustering, as shown by the reported increments in performance. The most significant improvements can be observed on Arabic and Telugu (the language with the lowest baseline scores among the ones included in the study). Excluding Korean, for which performance is extremely poor, the only two languages for which significant improvements were not seen, are Indonesian and Swahili, which are morphosyntactically most similar to English.

As shown in Table 2, the $\text{ICEBERT}_{\text{BASE}}$ model significantly outperforms the $\text{ICEBERT}_{\text{SMALL}}$ model, but its scores remain well below the fully trained $\text{mBERT}_{\text{BASE}}$ model, as reported by Hu et al. (2020). We hope that the performance improvements of ICEBERT tokenization scale to more competitive models as well, but the model sizes considered here were limited by the available resources. Our model is also limited in training only on nine languages. Including more languages is perfectly possible and would only affect clustering time. Finally, we only considered question answering in this study. We expect our conclusions to generalize to other tasks, but this remains an open question for now.

In the paper we have compared clustering of monolingual vocabularies to multilingual segmentation, i.e., BPE on concatenated corpora. Other alternatives have been proposed in the literature, including multilevel tokenization (Wang et al., 2021) and tokenization-free, character-based modeling (Clark et al., 2021). It is beyond the scope of this paper to systematically compare all these different approaches. The idea of using sequences of token group IDs to represent input sentences was previously explored in Wulff and Søgaard (2015), who used finite state automata instead of clusters.

6 Conclusion

This study aimed to improve multilingual language models by training them on clusters of monolingual segments. The proposed approach yielded good quality clusters able to group semantically similar words and subwords across languages. Our clustering strategy led to improvements over standard segmentation and training methods on the majority of the included languages, when evaluated on a question answering task.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Hyung Won Chung, Dan Garrette, Kiat Chuan Tan, and Jason Riesa. 2020. Improving multilingual models with language-clustered vocabularies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4536–4546, Online. Association for Computational Linguistics.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2021. Canine: Pre-training an efficient tokenization-free encoder for language representation.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*, pages 8440–8451. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Philipp Dufter and Hinrich Schütze. 2020. Identifying necessary elements for bert's multilinguality.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*.

Goran Glavas, Robert Litschko, Sebastian Ruder, and Ivan Vulic. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. *CoRR*, abs/1902.00508.

Hideki Hashimoto, Youhei Sonobe, and Mutsunori Yagiura. 2010. A multilevel scheme with adaptive memory strategy for multiway graph partitioning. In *Learning and Intelligent Optimization*, pages 188–191, Berlin, Heidelberg. Springer Berlin Heidelberg.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization.

George Karypis and Vipin Kumar. 1995. Multilevel graph partitioning schemes. In *ICPP (3)*, pages 113–122. CRC Press.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *CoRR*, abs/2002.11794.

Jindrich Libovický, Rudolf Rosa, and Alexander Fraser. 2019. How language-neutral is multilingual bert? *CoRR*, abs/1911.03310.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Sampo Pyysalo, Jenna Kanerva, Antti Virtanen, and Filip Ginter. 2020. Wikibert models: deep transfer learning for many languages.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *ICASSP*, pages 5149–5152. IEEE.

P.H. Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31:1–10.

Jasdeep Singh, Bryan McCann, Richard Socher, and Caiming Xiong. 2019. BERT is not an interlingua and the bias of tokenization. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 47–55, Hong Kong, China. Association for Computational Linguistics.

Xinyi Wang, Sebastian Ruder, and Graham Neubig. 2021. Multi-view subword regularization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 473–482, Online. Association for Computational Linguistics.

Julie Wulff and Anders Søgaard. 2015. Learning finite state word representations for unsupervised Twitter adaptation of POS taggers. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 162–166, Beijing, China. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *CoRR*, abs/2010.11934.