Improving Multilingual Models with Language-Clustered Vocabularies

Hyung Won Chung* **Dan Garrette** **Kiat Chuan Tan** **Jason Riesa**
Google Research
{hwchung,dhgarrette,kiatchuan,riesa}@google.com

Abstract

State-of-the-art multilingual models depend on vocabularies that cover all of the languages the model will expect to see at inference time, but the standard methods for generating those vocabularies are not ideal for massively multilingual applications. In this work, we introduce a novel procedure for multilingual vocabulary generation that combines the separately trained vocabularies of several automatically derived language clusters, thus balancing the trade-off between cross-lingual subword sharing and language-specific vocabularies. Our experiments show improvements across languages on key multilingual benchmark tasks TyDi QA (+2.9 F1), XNLI (+2.1%), and WikiAnn NER (+2.8 F1) and factor of 8 reduction in out-of-vocabulary rate, all without increasing the size of the model or data.

1 Introduction

Multilingual models such as mBERT (Devlin et al., 2019), XLM (Lample and Conneau, 2019), and XLM-R (Conneau et al., 2020) have built on the advances of deep contextualized language modeling by pretraining on texts from many languages at once. One trait common to all of these models is the use of a single vocabulary containing subwords from all languages, used to segment the input text before transforming it into a sequence of embeddings. Conneau et al. (2020) showed that increasing the vocabulary size can produce quality gains, but unlike similar monolingual models, the vocabulary embedding matrix in each of these multilingual models constitutes a significant fraction of its total parameters; for example, 47% of XLM-R's parameters are in its embedding matrix. Therefore, scaling up a model's vocabulary size requires the construction of inductive biases that will guide the training procedure to effectively and efficiently learn these parameters.

The multilingual subword vocabularies used by the state-of-the-art models are generated by algorithms such as WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016), SentencePiece (Kudo and Richardson, 2018),[1] or Byte Pair Encoding (BPE) (Sennrich et al., 2016). Given a desired vocabulary size, these algorithms select an inventory of subwords that compactly represents the training corpora, which means preferring subwords that occur frequently, and, by extension for multilingual models, occur frequently across languages.

Because these algorithms look at overall subword frequencies in the combined multilingual corpus, they may learn suboptimal decompositions for low-resource languages that happen to have character combinations that resemble subwords of high-resources languages.[2] Additionally, subwords in common scripts like Latin and Cyrillic have a higher chance of selection since their counts are combined across a large number of languages (Wu and Dredze, 2019).

By attempting to optimize for the best overall vocabulary across all languages without regard for the differences among those languages, the joint procedure over-emphasizes cross-lingual subword sharing—even across languages with little lexical overlap—which is at odds with the finding of K et al. (2020) that subword sharing is not the principal reason for the effectiveness of multilingual models. It also *under*-emphasizes the need for all languages—particularly low-resource languages or those written in scripts used by few languages—to contribute subwords that are most effective for their own representations.

*Work done as a member of the Google AI Residency Program.

[1] SentencePiece uses BPE or unigram language model.

[2] For example, the fact that *la* is a good subword in French and Spanish does not mean that the algorithm should split the prefix *la* away from words in all languages.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 4536–4546,
November 16–20, 2020.

In this paper, we propose a novel approach to multilingual vocabulary generation that seeks to balance the trade-off between optimizing for cross-lingual subword sharing and the need for robust representation of individual languages. At a high level, we: 1) automatically group languages into clusters based on the distributional similarities of their individual subword vocabularies, 2) apply the SentencePiece algorithm separately to the data for each individual cluster, and finally 3) combine all cluster-vocabularies together to form a single unified multilingual model vocabulary.

We evaluate our approach on three distinct downstream tasks: TyDi QA, XNLI, and WikiAnn NER. Our experimental results show that our method improves model performance on all three tasks, and achieves a state-of-the-art result for TyDi QA and zero-shot cross-lingual NER. Crucially, our method improves performance without any changes to the model, and since it does not depend on the model architecture, it can be applied to any model that uses a vocabulary.

2 Clustered Vocabularies

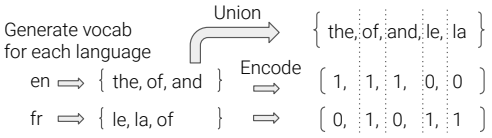

Figure 1: Encoding languages into binary vectors.

We begin by defining a vector representation for each language (Figure 1). For each language l in the set of languages L, we generate an l-specific vocabulary V^l by running the SentencePiece algorithm on l's corpus. Then we take the union of the resulting vocabulary to form the global vocabulary, $V^L = \bigcup_{l \in L} V^l$. Next, for each language, we form a binary vector $\mathbf{v}^l$ of dimension $\left|V^L\right|$, where each component of $\mathbf{v}^l$ corresponds to a subword in V^L, i.e., $\mathbf{v}^l_i = \mathbb{I}[V^L_i \in V^l]$ where $\mathbb{I}$ is an indicator function. In other words, $\mathbf{v}^l$ contains a 1 corresponding to each subword that is present in l's vocabulary, and two languages will have similar representations when their vocabularies have more subwords in common.

With each language encoded as a vector $\mathbf{v}^l$, we can apply standard clustering algorithms to assign each l to some c_j in the set of clusters C. For our experiments, we used k-means with cosine distance. Preliminary experiments (Table 1) indicated

k	1	4	8	16	32	104
TyDi QA	55.6/69.4	56.9/70.5	**58.5/71.7**	57.3/70.9	57.0/70.2	55.5/69.3

Table 1: TyDi QA results for our k-means-based vocabulary generation approach on different values of k. $k = 1$ puts all languages in a single cluster, and is thus equivalent to the baseline Joint approach; $k = 104$ generates a separate vocabulary for each language.

that using $k = 8$ would yield good results for our 104-language (the same set of languages used in mBERT) pretraining data so we fixed this value for all remaining experiments.

We then generate a vocabulary V^{c_j} for each cluster $c_j \in C$ by pooling all of the pretraining data for all languages $l \in c_j$ and running the SentencePiece algorithm. We set the target vocabulary size to be proportional to the size of the union of the individual vocabularies V^l of the languages belonging to the cluster i.e. $|V^{c_j}| \propto |\bigcup_{l \in c_j} V^l|$, ensuring that the proportion of the overall vocabulary allocated to a particular cluster is guided by two factors: it will increase if the cluster has more languages, and decrease if the cluster's languages have more vocabulary overlap. Finally, the multilingual model's overall vocabulary is the union of all of the cluster vocabularies: $V^C = \bigcup_{c_j \in C} V^{c_j}$.

Note that our method is a generalization of the conventional approach, which has $|C| = 1$.

3 Intrinsic Analysis

In this section, we directly examine the vocabularies produced by the standard recipe (Joint) and our clustering-based method (Cluster) in order to assess the degree to which each approach is able to capture and balance the inductive biases introduced in §1: a multilingual vocabulary should encourage subword sharing across languages when appropriate, but each language should also have the freedom to contribute the subwords that are most effective for its own representation.

For all of our experiments, we generate vocabularies from the Wikipedia articles of 104 languages with 906M sentences. Each multilingual vocabulary is 488k subwords. The cluster assignments and sizes generated by Cluster are found in Table 2.

In order to quantify the extent of subword sharing between languages, we look at each language as a *distribution* over the vocabulary. In particular, we apply the multilingual vocabulary V's segmentation to each language l's monolingual corpus in order to count the frequency of each sub-

Name	Clusters	$\lvert V^c \rvert$
c_1	af, sq, hy, az, bn, bs, my, ceb, hr, en, fi, ka, el, he hu, id, ga, jv, lv, lt, ms, min, pl, pa, sco, hbs, sl su, sw, tl, th, tr, uz, vi, cy, yo	200,306
c_2	ar, bpy, fa, azb, ur, lah	40,218
c_3	an, ast, eu, ca, gl, io, it, la, lmo, oc, pms pt, ro, scn, es, war	80,764
c_4	ba, be, bg, ce, cv, kk, ky, mk, mn, ru, sr, tg, tt, uk	82,163
c_5	bar, br, fr, de, ht, nds, lb, mg, vo	51,653
c_6	zh-Hans, zh-Hant, ja, ko	25,528
c_7	cs, da, nl, et, is, nb, nn, sk, sv, fy	57,681
c_8	gu, hi, kn, ml, mr, ne, new, ta, te	61,683

Table 2: Cluster definitions of CLUSTER.

JOINT	en	ur	ja	zh
en	0	8	9	15
ur		0	14	20
ja			0	6

CLUSTER	en	ur	ja	zh
en	0	145	306	307
ur		0	165	166
ja			0	1

Table 3: Wasserstein-1 distance ($\times 1000$).

word. The empirical distribution of a language l over the vocabulary V is defined by normalizing the frequencies of the monolingual corpus. Given these distributional representations of languages, we can use the Wasserstein-1 distance, W_1, between two languages to quantify the extent of subword sharing (Table 3). Unlike JOINT for which W_1 is relatively small even for languages with distinct scripts (e.g., English and Urdu), CLUSTER manifests much larger values for empirically different languages (and hence in different clusters) while having even smaller values for languages in the same cluster (e.g., Japanese and Chinese). This suggests that the clustering based approach not only minimizes the subword sharing when languages are dissimilar but it also puts more emphasis on subword sharing when necessary.

We quantify the degree of language freedom granted by each approach by examining the fraction of the vocabulary's subwords that contain rare scripts (Table 4). CLUSTER has a higher percentage of Chinese, Japanese and Korean (grouped together as CJK) subwords because CJK languages are in the same cluster and by themselves, so CJK subwords can be selected independent of other languages. A similar pattern exists for Arabic script.

4 Experiments

In order to demonstrate the effectiveness of our approach across languages and downstream tasks, we evaluate our method on three distinct datasets:

	CJK	Arabic
JOINT	2.8	3.4
CLUSTER	4.4	6.1

Table 4: Percentage of each vocabulary's subwords that contain CJK, or Arabic script characters.

- TYDI QA (Clark et al., 2020): question answering in 10 languages. Results in Table 5.
- XNLI (Conneau et al., 2018): natural language inference in 15 languages. Results in Table 6.
- WikiAnn NER (Pan et al., 2017): named entity recognition in 40 languages. Results in Table 7.

The principal goal of this work is to investigate the effect of improved *vocabulary composition* on multilingual models. We make a good-faith effort to control all other variables, e.g., hyperparameters, training/evaluation procedures. In particular, we keep the number of languages constant, since per-language model capacity is known to affect the performance of multilingual models as shown in Conneau et al. (2020). In addition, we keep the number of parameters constant[3], including the vocabulary size, since the performance of Transformer-based (Vaswani et al., 2017) models is strongly correlated with number of parameters (Lepikhin et al., 2020; Kaplan et al., 2020; Raffel et al., 2020; Conneau et al., 2020; Brown et al., 2020). With carefully controlled experiments, we are confident that any improvement in the results is solely attributable to vocabulary composition. In other words, our vocabularies improve models without increasing the model size, compute or data.

We pretrain using the masked language modeling (MLM) task on the raw text without applying any language-specific pre-processing.[4]

At a high level, the experimental results demonstrate that our CLUSTER approach to multilingual vocabulary generation improves over the standard JOINT recipe, increasing the macro average performance across languages for all datasets and tasks. For TYDI QA, we see improvements on all languages. For XNLI, CLUSTER perform better or equally well in all languages except French. The results on individual languages are more mixed for NER, with CLUSTER providing large gains on low-resource or rare-script languages, and small losses on high-resource languages.

In the remainder of this section, we provide more

[3]The only exception is the full scale model in §4.2.

[4]See Appendix A for additional training details.

	(en)	ar	bn	fi	id	ja	ko	ru	sw	te	th	**Avg**
MINSPAN												
JOINT	48.4	70.4	61.5	54.0	55.4	42.6	40.6	48.2	53.6	75.8	54.0	55.6
CLUSTER	**50.3**	**71.9**	**64.8**	**57.3**	**58.0**	**47.2**	**45.5**	**49.4**	**57.0**	**77.5**	**56.1**	**58.5**
SELECTP												
JOINT	63.7	83.6	71.5	65.5	65.8	55.8	63.3	67.7	66.8	84.9	69.4	69.4
CLUSTER	**65.2**	**85.7**	**72.2**	**68.8**	**69.6**	**59.2**	**65.5**	**67.9**	**71.3**	**86.1**	**70.7**	**71.7**

Table 5: Results on the TYDI QA primary tasks: minimal answer span (MINSPAN) and passage selection (SELECTP). The final column (**Avg**) is the macro average *excluding English*, following Clark et al. (2020).

	ar	bg	de	el	en	es	fr	hi	ru	sw	th	tr	ur	vi	zh	**Avg**
JOINT	66.6	70.8	72.1	66.9	**81.3**	74.4	**74.1**	63.6	70.8	54.1	64.6	65.4	60.6	68.3	66.4	68.0
CLUSTER	**69.1**	**71.9**	**72.5**	**72.6**	**81.3**	**75.4**	74.0	**68.0**	**71.1**	**56.7**	**67.6**	**66.9**	**63.7**	**69.6**	**71.0**	**70.1**

Table 6: XNLI accuracies.

in-depth analyses of these results (§4.1), and show that our approach continues to perform well when used to train a large scale model (§4.2).

4.1 Analysis

We use the *minimum description length* principle (MDL) (Rissanen, 1989) to aid in our analysis. Consider an example where we want to encode data with a codebook. For example, each word in the data can be encoded by a unique sequence of bits, which is referred to as a codeword. The MDL principle favors the codebook with the minimal description length. Following Goldwater (2007), we define the description length (DL) as the length of combined codebook and encoded data; that is, the sum of the lengths of all codewords in our codebook plus the length of the encoded data.

We apply this to the setting of learning a subword vocabulary for neural network models. Our codebook is a mapping from a subword to a unique integer index of fixed length, typically 32 bits. Therefore, the description length is equivalent to the sum of the number of unique subwords, i.e. the vocabulary size, and the number of integers of the encoded or tokenized input data. Comparing the description length of two vocabularies is therefore equivalent to comparing the total number of subwords after the input corpus is tokenized by each vocabulary. Without loss of generality, we use the average number of tokens per sentence as an equivalent measure of the description length.

As shown in Table 8, CLUSTER does indeed reduce the DL of the training corpus, which correlates with the performance improvements we see on downstream tasks. With smaller DL, the input text is encoded with *longer* subwords, which we might expect to lead to a higher out-of-vocabulary (OOV) rate (Arivazhagan et al., 2019). However, CLUSTER has an 8 times smaller OOV rate than JOINT (Table 8). We believe that this is related to the larger extent of language freedom as evidenced by the particularly large OOV rate reductions observed in rare-script languages. For example, the OOV rate is reduced by a factor of 26 (Korean), 18 (Japanese) and 17 (Chinese).

The longer DL of JOINT means the average subword length is shorter. As a result, the model has learn to map from finer-grained input sequences to semantic meaning (Arivazhagan et al., 2019). As an extreme example, a character-based model would have to learn how to *reconstruct* each word, while a word-based model is exempt from this task. Though the difference in DL is smaller than this extreme case, the same logic applies and JOINT must learn a more complex function than CLUSTER.

Finally, we note that this correlation between DL and downstream performance means we can use DL as a proxy metric to compare vocabularies, allowing for faster iteration over various vocabulary generation approaches without having to run the expensive model training.

Low-resource languages with common scripts. Languages like Swahili and Yoruba use Latin script but have small amounts of data and CLUSTER outperforms JOINT for all three tasks on these languages, which we believe is attributable to better segmentation. In §1, we highlighted one example where over-segmentation can occur,[5] but we

[5]While JOINT segments the common Swahili adverb *lazima* into *la* and *zima*, CLUSTER keeps it intact.

	c_1				c_2			c_3		c_4		c_5		c_6	c_7		c_8				Avg
	en	sw	yo	he	ar	fa	ur	es	eu	bg	ru	fr	de	ko	nl	et	hi	ml	ta	te	
JOINT	**84.5**	64.7	54.8	52.0	48.9	36.4	32.9	**77.4**	**54.5**	**80.4**	**65.5**	79.9	78.3	50.7	**82.3**	75.6	64.5	56.6	58.4	48.1	61.7
CLUSTER	84.1	**72.2**	**62.8**	**59.2**	**53.5**	**51.0**	**61.8**	75.5	53.1	78.6	62.0	**81.8**	**78.4**	**56.8**	81.8	**77.0**	**70.8**	**63.2**	**60.9**	**56.8**	**64.5**

Table 7: Results for zero-shot NER using cross-lingual transfer from English (following XTREME (Hu et al.)) for a sample of languages, grouped according to the clustering used by CLUSTER. All scores are labeled span F1, and **Avg** is the macro average across all 40 XTREME languages.

	Avg. DL	OOV rate [%]
JOINT	20.9	0.200
CLUSTER	**20.2**	**0.025**

Table 8: Average description length and OOV rate.

Model	**TYDI QA**	**XNLI**	**NER**
mBERT	52.7/64.4	65.4	62.2
XLM-R	-	**79.2**	65.4
Ours	**63.4/77.7**	77.0	**73.6**
Human	70.1/79.9	92.8	-

Table 9: Comparisons with a full-scale model using our approach. Scores are macro averages over languages. TYDI QA numbers are in (MINSPAN/SELECTP) format, and our results are on the test set via a submission to the official leaderboard. Baseline numbers for TYDI QA are from Clark et al. (2020) and the rest are from Hu et al..

can quantify this analysis more generally with DL: CLUSTER has 9.4% (Swahili) and 11.1% (Yoruba) shorter DL compared to JOINT, and this matches well with 7.5 and 8.0 higher NER F1, respectively.

Rare-script languages. For languages with rare scripts (e.g., CJK and Thai), CLUSTER strongly outperforms JOINT in all tasks. For NER, particularly large gains are achieved for Arabic-script languages in cluster c_2 (e.g., 28.9 F1 improvement in Urdu) and Indian languages in c_8. We hypothesize that the gain for this group of languages is due to the higher coverage of subwords in rare scripts, and consequently lower OOV rates (Table 4).

On clusters with languages in different scripts. Our vocabularies were trained from the Wikipedia articles, which frequently contain translations/transliterations. For example, the first sentence of both the Hindi- and Tamil-language versions of the article on "India" contain the exact English phrase "Republic of India". Since many of the same people/places will be topics in articles across Indic-script languages, it is not too surprising that the clustering algorithm groups some of these languages. We note that the NER performance on the Indic languages in cluster c_8 are especially strong (Table 7) possibly due to such shared representation of named entities.

A similar phenomenon can be seen with Korean, since Chinese characters are often appended in parentheses to a Korean word to disambiguate polysemy, which is particularly common for formal contents like Wikipedia. This is why Chinese, Japanese (having large lexical overlap with Chinese) and Korean are in the same cluster c_6. We note that these languages show especially strong improvement in all three tasks we considered as well as drastically large reduction in OOV rate.

Therefore, we see these behaviors as a *strength* of our data-driven approach, flexibly capturing the characteristics of the data which may not be obvious from a purely linguistic perspective.

4.2 Full-scale model

To evaluate the effectiveness of our method on a large scale model, we train a 24-layer Transformer model with CLUSTER and report the macro-averaged results in Table 9. We drastically outperform the baseline for TYDI QA with about 10.7 F1 absolute improvement on MINSPAN and 13.3 F1 on SELECTP, and NER with 8.2 F1 absolute improvement over XLM-R. There is a small loss on XNLI, though this may be due to training on less data, and only Wikipedia domain.

5 Conclusion

We describe a novel clustering-based multilingual vocabulary generation algorithm. We showed that this empirically motivated clustering-based method consistently outperforms the standard vocabulary generation recipe used by most multilingual pretrained language modeling work.

Acknowledgements

We would like to thank Vera Axelrod, Tim Dozat, Melvin Johnson, Thibault Févry, Xavier Garcia, and Karthik Raman for their feedback on this work.

References

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively Multilingual Neural Machine Translation in the Wild: Findings and Challenges. *arXiv e-prints*, page arXiv:1907.05019.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCand lish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv e-prints*, page arXiv:2005.14165.

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. *Transactions of the Association for Computational Linguistics*.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sharon J. Goldwater. 2007. *Nonparametric Bayesian Models of Lexical Acquisition*. Ph.D. thesis, Brown University.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalization. In *International Conference on Machine Learning, ICML 2020,*.

Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. 2020. Cross-Lingual Ability of Multilingual BERT: An Empirical Study. In *Proc. of the International Conference on Learning Representations*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv e-prints*, page arXiv:2001.08361.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. *arXiv e-prints*, page arXiv:2006.16668.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. 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.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association*

for Computational Linguistics, pages 151–164, Florence, Italy. Association for Computational Linguistics.

Jorma Rissanen. 1989. *Stochastic Complexity In Statistical Inquiry*.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. 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 833–844, Hong Kong, China. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv e-prints*, page arXiv:1609.08144.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*.

A Experiment details

A.1 Pretraining

We did not use any hyperparameter search for pretraining. We use LAMB optimizer (You et al., 2020) with batch size of 4096. You et al. (2020) also recommended learning rate of 0.0018, warm-up proportion of 2.5% of the total number of steps. We use linear warm-up and linear learning rate decay down to 0 in the last step. We use gradient clipping with a norm of 1.0.

For all the experiments except for the full-scale model, we trained using 64 Google Cloud TPUs. We pretrained models with two different sequence lengths: 128 and 512 with 500k and 125k steps, respectively. The model with sequence length of 512 runs at about 2.3 steps/second, which takes about 16 hours to finish; this model was used to run WikiAnn NER and TyDi QA. The model with sequence length of 128 runs at 9.8 steps/second and finishes in about 15 hours; this model was used to run XNLI. A step refers to one gradient update of the LAMB optimizer.

During pretraining, we sampled each language's data with the following strategy. First we compute the empirical distribution for each language

$$p^l = \frac{n^l}{\sum_{l' \in L} n^{l'}} \quad (1)$$

where n^l is the number of sentences in language l's corpus. Then we use the exponential smoothing value of 0.7 following (Devlin et al., 2019), i.e., we exponentiate p^l and renormalize to compute the sampling probabilities of each language.

We used whole-word masking during pretraining. However, since some languages do not typically use whitespace between words (e.g., Thai), we used the heuristic of SentencePiece meta symbol U+2581 to designate the beginning of the word. Therefore, a word is defined as the token span between two successive U+2581 symbols.

For the full-scale model, we pretrained with 256 Google Cloud TPUs for 1.5 million steps with a batch size of 4096 and learning rate of 0.0018, which took 8 days. We only trained one model, and used a sequence length of 512.

All experiments were run in TensorFlow.

A.2 SentencePiece configurations

We used the following configuration to train a SentencePiece model: unigram language model, character coverage of 0.9995, and 1M seed sentences.

A.3 Model architecture

Except for the full-scale model, all models were 12 layers of Transformers, with hidden size of 768 and 12 attention heads. For faster experimentation, we used an embedding size of 128, similar to Lan et al. (2020). The total number of parameters is 150M. The full-scale model has 24 Transformer layers, hidden size of 1024, and 16 attention heads. We used an embedding size of 512, totaling 550M parameters. We chose this number of parameters to mimic XLM-R.

A.4 Fine-tuning

We ran experiments with two seed values and chose the best model based on the average of the two runs. For fine-tuning, we used Adam optimizer (Kingma and Ba, 2015).

For WikiAnn NER, we used a learning rate of 4×10^{-5}, a batch size of 32, and 2 training epochs. We found that the performance is robust with respect to the set of hyperparameters, so we did not change this setting. The training was run with 4 TPUs which took about one hour to finish. The evaluation metric is span-level F1 score. Our evaluation code was tested against the `seqeval` library: https://github.com/chakki-works/seqeval, and produces the same scores.

For XNLI, we used a batch size of 32, performed grid search over the learning rate of $[1 \times 10^{-5},\ 2 \times 10^{-5},\ 3 \times 10^{-5}]$, and trained for 3 epochs. We chose the best model on the development set based on the macro-averaged accuracy and then used that model to report on the test set. The training was run with 8 TPUs which took about 2-3 hours to finish. The evaluation metric is classification accuracy.

For TyDi QA, we found that larger batch sizes improved the training stability, so we used a batch size of 512. With the larger batch size, longer epochs were helpful, so we used a grid search over the learning rate of $[3 \times 10^{-5},\ 4 \times 10^{-5},\ 5 \times 10^{-5}]$ and training epochs of $[7,\ 8,\ 9]$. We chose the best model based on the macro-averaged F1 score on 10 languages, excluding English following Clark et al. (2020). For the hyperparameters that are specific to TyDi QA, we used the same settings as the baseline model from (Clark et al., 2020): 45 maximum passages, 0.1 include unknown rates, sequence length of 512, and window stride of 128. The evaluation metric is F1 score, which we computed with the official evaluation script from https://github.com/google-research-

	ar	bg	de	el	en	es	fr	hi	ru	sw	th	tr	ur	vi	zh	**Avg**
JOINT	66.9	70.2	72.9	67.0	**81.4**	75.2	74.5	63.3	**71.0**	53.5	63.0	66.1	60.9	68.5	67.4	68.1
CLUSTER	**68.3**	**71.7**	**73.5**	**74.1**	81.0	**75.4**	**75.3**	**67.4**	70.6	**57.1**	**67.8**	**66.9**	**62.9**	**69.4**	**70.9**	**70.2**

Table 10: XNLI accuracy on development set. The best performing hyperparameters for both JOINT and CLUSTER were learning rate of 2×10^{-5}, batch size of 32, and 3 training epochs.

	c_1				c_2			c_3		c_4		c_5		c_6	c_7		c_8				**Avg**
	en	sw	yo	he	ar	fa	ur	es	eu	bg	ru	fr	de	ko	nl	et	hi	ml	ta	te	
JOINT	**84.3**	64.8	51.9	51.7	49.0	36.3	31.7	**76.7**	**53.3**	**81.0**	**66.4**	79.9	78.2	51.1	**81.9**	76.0	63.4	53.4	59.2	49.1	61.3
CLUSTER	84.1	**72.1**	**64.1**	**58.8**	**54.2**	**50.9**	**61.8**	74.7	51.8	79.3	63.0	**81.5**	**78.6**	**57.3**	81.2	**76.9**	**72.1**	**61.4**	**61.9**	**56.0**	**64.3**

Table 11: WikiAnn NER F1 scores on development set. We used 4×10^{-5}, batch size of 32 and 2 training epochs for both CLUSTER and JOINT.

datasets/tydiqa.

B Datasets

For pretraining, we use 906M sentences of Wikipedia data covering 104 languages. The number of examples of the three datasets used for evaluation is summarized in Table 15. The XNLI data has a training set in English and development and test sets in 15 languages, which can be downloaded from https://cims.nyu.edu/ sbowman/multinli/.

The TYDI QA datasets are in 11 languages including English, which is excluded from the official evaluation. The link to download the dataset is https://github.com/google-research-datasets/tydiqa.

For Wikiann NER data, we follow XTREME (Hu et al.) and used the balanced train, development, test set splits of Rahimi et al. (2019), for 40 languages. The dataset can be downloaded from https://github.com/afshinrahimi/mmner.

We did not exclude any examples from the three datasets. For Wikiann NER, Greek, Thai, Japanese, Korean and Chinese data have incorrect IOB2 encoding, e.g., `I-PER` following `O`. We fixed those encoding with a simple rule such that a tag with `I` prefix starting after `O` is corrected to have `B` prefix.

We did not use any preprocessing for XNLI or TYDI QA.

C Performance on development set

The main body of the paper contains test set results for XNLI and WikiAnn NER. In this section, we report the development set results so that researchers can try to reproduce the results without consulting to the test set.

Table 10 shows the results for XNLI and the results for WikiAnn NER are summarized in Table 11. Both tables contain the best performing hyperparameters in the caption. Since the TYDI QA test set is private, we performed all experiments on the development set for TYDI QA except for the full-scale model for which we submitted to the official leaderboard.

C.1 Development set results for the full-scale model

For the full-scale model, Table 14 shows the development set results on WikiAnn NER, Table 13 for XNLI and, Table 12 for TYDI QA. We found that this large Transformer model requires different sets of hyperparameters to be effective. We used the LAMB optimizer to match the pretraining since it made the training more stable. For XNLI we did a grid search on over learning rates $[9 \times 10^{-5},\ 1 \times 10^{-4}]$, training epochs $[9,\ 10,\ 11]$, and a fixed batch size of 512. For TYDI QA we did a grid search over learning rates $[8 \times 10^{-5},\ 9 \times 10^{-5}]$ and training epochs $[13,\ 14,\ 15]$, and a fixed batch size of 512. For WikiAnn NER, we chose the learning rate from $[2 \times 10^{-5},\ 3 \times 10^{-5}]$, and used a fixed batch size of 32 and 2 training epochs.

C.2 WikiAnn NER test set results on all 40 languages

In this section, we expand the results in and Table 7 and Table 9, which only show subset of languages (or only average for the latter.

For these test set results, Table 16 expands on Table 7 and show results on all 40 languages considered in XTREME. Table 17 expands on Table 9 in a similar manner.

C.3 Language information

Table 18 lists all 104 languages considered in this paper and their script information.

	ar	bn	fi	id	ja	ko	ru	sw	te	th	**Avg**
MINSPAN	77.0	72.1	66.1	65.1	52.9	55.8	57.2	65.9	82.3	59.7	65.4
SELECTP	88.5	81.3	76.7	76.0	66.1	72.7	74.3	80.1	90.6	73.8	78.0

Table 12: Full-scale model's result on the TYDI QA primary tasks (development set).

ar	bg	de	el	en	es	fr	hi	ru	sw	th	tr	ur	vi	zh	**Avg**
77.1	79.9	81.4	80.9	86.9	82.9	82.0	75.5	79.2	59.4	74.4	73.5	69.2	77.8	77.2	77.2

Table 13: XNLI (development set) result for the full-scale model. The best performing model has learning rate of 0.0001, train epochs of 9 and batch size of 512.

	c_1				c_2			c_3		c_4		c_5		c_6	c_7		c_8				**Avg**
	en	sw	yo	he	ar	fa	ur	es	eu	bg	ru	fr	de	ko	nl	et	hi	ml	ta	te	
CLUSTER	86.4	73.4	79.1	72.6	73.2	75.4	82.3	85.7	68.2	88.4	79.8	87.8	84.0	68.2	87.4	85.3	79.4	73.9	73.2	67.2	73.7

Table 14: WikiAnn NER F1 scores of the full-scale model on development set. The best performing model has learning rate of 3×10^{-5}, batch size of 32 and trained for 2 epochs.

	Training	Development	Test
XNLI	392,702	2490	5010
TYDI QA	166916	18670	18751
WikiAnn NER	20000	262300	262300

Table 15: Number of examples in training, development, test set splits for each evaluation datasets.

en	af	ar	bg	bn	de	el	es	et	eu	fa	fi	fr	he	hi	hu	id	it	ja	jv
86.2	84.0	73.1	88.1	84.0	84.0	82.8	86.7	85.3	68.8	76.2	84.5	87.5	72.9	77.7	83.0	65.1	85.0	24.9	68.6

ka	kk	ko	ml	mr	ms	my	nl	pt	ru	sw	ta	te	th	tl	tr	ur	vi	yo	zh
78.8	54.0	68.0	73.5	73.8	78.0	66.4	87.6	86.5	79.5	72.7	72.1	66.0	2.3	80.8	83.3	78.6	79.1	80.2	34.1

Table 16: WikiAnn NER results in Table 7 on all 40 languages. The average F1 scores are 61.7 and 64.5 for JOINT and CLUSTER, respectively.

en	af	ar	bg	bn	de	el	es	et	eu	fa	fi	fr	he	hi	hu	id	it	ja	jv
86.2	84.0	73.1	88.1	84.0	84.0	82.8	86.7	85.3	68.8	76.2	84.5	87.5	72.9	77.7	83.0	65.1	85.0	24.9	68.6

ka	kk	ko	ml	mr	ms	my	nl	pt	ru	sw	ta	te	th	tl	tr	ur	vi	yo	zh
78.8	54.0	68.0	73.5	73.8	78.0	66.4	87.6	86.5	79.5	72.7	72.1	66.0	2.3	80.8	83.3	78.6	79.1	80.2	34.1

Table 17: WikiAnn NER results in Table 9 on all 40 languages. The average F1 score is 73.6.

Language	Language code	Script	Language	Language code	Script
Afrikaans	af	Latin	Latvian	lv	Latin
Albanian	sq	Latin	Lithuanian	lt	Latin
Arabic	ar	Arabic	Lombard	lmo	Latin
Aragonese	an	Latin	Low-saxon	nds	Latin
Armenian	hy	Armenian	Luxembourgish	lb	Latin
Asturian	ast	Latin	Macedonian	mk	Cyrillic
Azerbaijani	az	Latin	Malagasy	mg	Latin
Bashkir	ba	Cyrillic	Malay	ms	Latin
Basque	eu	Latin	Malayalam	ml	Malayalam
Bavarian	bar	Latin	Marathi	mr	Devanagari
Belarusian	be	Cyrillic	Minangkabau	min	Latin
Bengali	bn	Bengali	Mongolian	mn	Cyrillic
Bishnupriya-manipuri	bpy	Bengali	Nepali	ne	Devanagari
Bosnian	bs	Latin	Newar	new	Devanagari
Breton	br	Latin	Norwegian-bokmal	nb	Latin
Bulgarian	bg	Cyrillic	Norwegian-nynorsk	nn	Latin
Burmese	my	Myanmar	Occitan	oc	Latin
Catalan	ca	Latin	Persian	fa	Arabic
Cebuano	ceb	Latin	Piedmontese	pms	Latin
Chechen	ce	Cyrillic	Polish	pl	Latin
Chinese-simplified	zh-Hans	Chinese	Portuguese	pt	Latin
Chinese-traditional	zh-Hant	Chinese	Punjabi	pa	Gurmukhi
Chuvash	cv	Cyrillic	Romanian	ro	Latin
Croatian	hr	Latin	Russian	ru	Cyrillic
Czech	cs	Latin	Scots	sco	Latin
Danish	da	Latin	Serbian	sr	Cyrillic
Dutch	nl	Latin	Serbo-croatian	hbs	Latin
English	en	Latin	Sicilian	scn	Latin
Estonian	et	Latin	Slovak	sk	Latin
Finnish	fi	Latin	Slovenian	sl	Latin
French	fr	Latin	South-azerbaijani	azb	Arabic
Galician	gl	Latin	Spanish	es	Latin
Georgian	ka	Georgian	Sundanese	su	Latin
German	de	Latin	Swahili	sw	Latin
Greek	el	Greek	Swedish	sv	Latin
Gujarati	gu	Gujarati	Tagalog	tl	Latin
Haitian	ht	Latin	Tajik	tg	Cyrillic
Hebrew	he	Hebrew	Tamil	ta	Tamil
Hindi	hi	Devanagari	Tatar	tt	Cyrillic
Hungarian	hu	Latin	Telugu	te	Telugu
Icelandic	is	Latin	Thai	th	Thai
Ido	io	Latin	Turkish	tr	Latin
Indonesian	id	Latin	Ukrainian	uk	Cyrillic
Irish	ga	Latin	Urdu	ur	Arabic
Italian	it	Latin	Uzbek	uz	Latin
Japanese	ja	Japanese	Vietnamese	vi	Latin
Javanese	jv	Latin	Volapuk	vo	Latin
Kannada	kn	Kannada	Waray-waray	war	Latin
Kazakh	kk	Cyrillic	Welsh	cy	Latin
Kirghiz	ky	Cyrillic	West	fy	Latin
Korean	ko	Korean	Western-punjabi	lah	Arabic
Latin	la	Latin	Yoruba	yo	Latin

Table 18: List of languages used in the pre-training.