Byte Pair Encoding is Suboptimal for Language Model Pretraining

Kaj Bostrom and **Greg Durrett**
Department of Computer Science
The University of Texas at Austin
{kaj,gdurrett}@cs.utexas.edu

Abstract

The success of pretrained transformer language models (LMs) in natural language processing has led to a wide range of pretraining setups. In particular, these models employ a variety of subword tokenization methods, most notably byte-pair encoding (BPE) (Sennrich et al., 2016; Gage, 1994), the WordPiece method (Schuster and Nakajima, 2012), and unigram language modeling (Kudo, 2018), to segment text. However, to the best of our knowledge, the literature does not contain a direct evaluation of the impact of tokenization on language model pretraining. We analyze differences between BPE and unigram LM tokenization, finding that the latter method recovers subword units that align more closely with morphology and avoids problems stemming from BPE's greedy construction procedure. We then compare the fine-tuned task performance of identical transformer masked language models pretrained with these tokenizations. Across downstream tasks and two languages (English and Japanese), we find that the unigram LM tokenization method matches or outperforms BPE. We hope that developers of future pretrained LMs will consider adopting the unigram LM method over the more prevalent BPE.

1 Introduction

Large transformers (Vaswani et al., 2017) pretrained with variants of a language modeling objective, such as BERT (Devlin et al., 2019), have proven their effectiveness at flexibly transferring to a variety of domains and tasks. One design decision that makes them particularly adaptable is their graceful handling of the open vocabulary problem through subword tokenization. Subword tokenization, popularized in the neural machine translation literature (Sennrich et al., 2016; Vaswani et al., 2017; Wu et al., 2016), produces tokens at multiple levels of granularity, from individual characters to full words. As a result, rare words are broken down into a collection of subword units, bottoming out in characters in the worst case.

Critically, a pretrained language model's subword vocabulary cannot be altered: any downstream application of these models must tokenize input or generate output using the original subword vocabulary, making the choice of tokenization a particularly significant decision.

A variety of subword tokenization methods have seen use in pretrained language models. BERT uses the WordPiece method (Schuster and Nakajima, 2012), a language-modeling based variant of BPE; T5 (Raffel et al., 2019) uses character-level BPE; GPT2 (Radford et al., 2019) and RoBERTa (Liu et al., 2019) use BPE over raw bytes instead of unicode characters; XLNet (Yang et al., 2019) and ALBERT (Lan et al., 2019) use the SentencePiece library (Kudo and Richardson, 2018) which implements both BPE and unigram language model tokenization, but in both cases fail to clarify which of these methods they chose. The effects of tokenization are not examined in a reported experiment in any of the above works except Liu et al. (2019), who note that WordPiece gave a small advantage over BPE in their preliminary investigation. In the machine translation literature, Kudo (2018) introduced the unigram language model tokenization method in the context of machine translation and found it comparable in performance to BPE. Domingo et al. (2018) performed further experiments to investigate the effects of tokenization on neural machine translation, but used a shared BPE vocabulary across all experiments. Gallé (2019) examined algorithms in the BPE family, but did not compare to unigram language modeling.

In this work, we characterize the space of proposed subword tokenization algorithms and analyze the differences between the two methods with

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4617–4624
November 16 - 20, 2020. ©2020 Association for Computational Linguistics

publicly available implementations: BPE (merging tokens based on bigram frequency) and unigram language modeling (pruning tokens based on unigram LM perplexity). While the vocabularies resulting from these schemes are heavily overlapping, we compare each method to reference morphological segmentations and find that the unigram LM method produces tokens better aligned with morphology. To understand whether this more natural tokenization leads to improved performance, we pretrain separate language models using the RoBERTa objective (Liu et al., 2019) with each tokenization for both English and Japanese, two typologically distant languages. On downstream tasks, we find a performance gap across tasks and languages, with the unigram LM method providing an improvement over BPE of up to 10% in our Japanese QA experiments, indicating the benefits of adopting this technique in the context of language model pretraining.

2 Algorithms

Subword tokenization algorithms consist of two components: a vocabulary construction procedure, which takes a corpus of text and returns a vocabulary with the desired size, and a tokenization procedure, which takes the built vocabulary and applies it to new text, returning a sequence of tokens. In theory, these two steps can be independent, although for the algorithms we examine the tokenization procedure is tightly coupled to the vocabulary construction procedure.

A BPE vocabulary is constructed as follows:

Algorithm 1 Byte-pair encoding (Sennrich et al., 2016; Gage, 1994)

Input: set of strings D, target vocab size k
procedure BPE(D, k)
  $V \leftarrow$ all unique characters in D
    (about 4,000 in English Wikipedia)
  while $|V| < k$ **do** ▷ Merge tokens
    $t_L, t_R \leftarrow$ Most frequent bigram in D
    $t_{\text{NEW}} \leftarrow t_L + t_R$ ▷ Make new token
    $V \leftarrow V + [t_{\text{NEW}}]$
    Replace each occurrence of t_L, t_R in
      D with t_{NEW}
  end while
  return V
end procedure

BPE tokenization takes the vocabulary V containing ordered merges and applies them to new text in the same order as they occurred during vocabulary construction.

The WordPiece algorithm (Schuster and Nakajima, 2012), used to construct BERT's vocabulary, closely resembles BPE. However, instead of merging the most frequent token bigram, each potential merge is scored based on the likelihood of an n-gram language model trained on a version of the corpus incorporating that merge. Schuster and Nakajima (2012) note that the process of estimating language model parameters for every potential merge is prohibitive, so they employ aggressive heuristics to reduce the number of potential merges considered. As their implementation is not public,[1] we are unable to make a comparison to this method.

The unigram LM method (Kudo, 2018), in contrast to the bottom-up construction process of BPE and WordPiece, begins with a superset of the final vocabulary, pruning it to the desired size:

Algorithm 2 Unigram LM (Kudo, 2018)

Input: set of strings D, target vocab size k
procedure UnigramLM(D, k)
  $V \leftarrow$ all substrings occurring more than
    once in D (not crossing words)
  while $|V| > k$ **do** ▷ Prune tokens
    Fit unigram LM θ to D
    for $t \in V$ **do** ▷ Estimate token 'loss'
      $L_t \leftarrow p_\theta(D) - p_{\theta'}(D)$
      where θ' is the LM without token t
    end for
    Remove $\min(|V| - k, \lfloor \alpha |V| \rfloor)$ of the
    tokens t with highest L_t from V,
    where $\alpha \in [0, 1]$ is a hyperparameter
  end while
  Fit final unigram LM θ to D
  return V, θ
end procedure

Unigram LM tokenization takes the vocabulary V and unigram LM parameters θ and performs Viterbi inference to decode the segmentation with maximum likelihood under θ. This method is similar to Morfessor's unsupervised segmentation (Creutz and Lagus, 2005) without its informed prior over token length.

[1] Although its name and association with Google might suggest otherwise, the SentencePiece library (Kudo and Richardson, 2018) does not, in fact, implement the WordPiece algorithm; it provides implementations of BPE and unigram LM based tokenization.

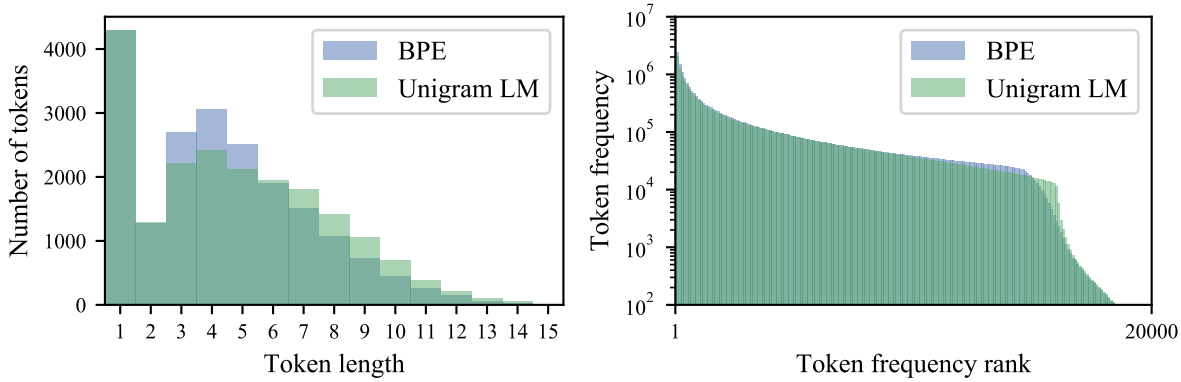
Original:	furiously	**Original:**	tricycles	**Original:**	nanotechnology
BPE:	_fur iously	**BPE:**	_t ric y cles	**BPE:**	_n an ote chn ology
Uni. LM:	_fur ious ly	**Uni. LM:**	_tri cycle s	**Uni. LM:**	_nano technology

Original: Completely preposterous suggestions
BPE: _Comple t ely _prep ost erous _suggest ions
Unigram LM: _Complete ly _pre post er ous _suggestion s

Original:	corrupted	**Original:**	1848 and 1852,
BPE:	_cor rupted	**BPE:**	_184 8 _and _185 2,
Unigram LM:	_corrupt ed	**Unigram LM:**	_1848 _and _1852 ,

Original	磁性は様々に分類がなされている。						
BPE	磁	性は	様々	に分類		がなされている	。
Unigram LM	磁	性	は	様々	に	分類	がなされている	。
Gloss	magnetism	(top.)	various ways	in	classification	is done	.
Translation	Magnetism is classified in various ways.						

Figure 1: Example tokenizations. The character '_' is a word boundary marker. BPE merges common tokens, such as English inflectional suffixes and Japanese particles, into their neighbors even when the resulting unit is not semantically meaningful.



(a) Token length distributions within each vocabulary

(b) Token frequency profiles over the corpus

Figure 2: English subword vocabulary and corpus profiles. The unigram LM method produces longer tokens on average (a) and uses its vocabulary space more effectively (b), with more tokens of moderate frequency.

In the course of our experiments we did not observe a major difference in speed between the two algorithms. Both require similar amounts of time to construct a vocabulary, and both have a negligible impact on overall model inference latency.

3 Comparison of Segmentations

3.1 Morphology

In Figure 1 we illustrate the differences in tokenization output between BPE and the unigram LM method. We observe that the unigram LM method produces subword units that qualitatively align with morphology much better than those produced by BPE. In particular, we note that the unigram LM method recovers common affixes such as *-ly*, *-s*, *pre-*, and *tri-* while BPE does not, instead absorbing them into adjacent units (*-cles*) while also producing meaningless single-character units.

This trend is supported by Table 1, in which

More frequent in									
BPE					Unigram LM				
_H	_L	_M	_T	_B	s	.	,	ed	d
_P	_C	_K	_D	_R	ing	e	ly	t	_a

Table 1: Tokens with the highest difference in frequency between tokenizations. The unigram LM method tends to produce more parsimonious prefixes and suffixes.

	Tokenization	
	BPE	Unigram LM
Tokens per word type	4.721	4.633
Tokens per word	1.343	1.318

Table 2: Mean subword units per word for each method across all of English Wikipedia.

we observe that recognizable affixes appear much more frequently in the unigram LM tokenization of our pretraining corpus than in the BPE tokenization.

Method	English (w.r.t. CELEX2)			Japanese (w.r.t. MeCab)		
	Precision	Recall	F1	Precision	Recall	F1
BPE	38.6%	12.9%	19.3%	78.6%	69.5%	73.8%
Uni. LM	62.2%	20.1%	30.3%	82.2%	72.8%	77.2%

Table 3: Correspondence of subword boundaries between unsupervised tokenization methods and morphological reference segmentations.

As the BPE tokenization is constructed greedily according to frequency, common affixes (and punctuation) are frequently absorbed into other tokens.[2]

We see in Figure 2a that the unigram LM tokenization tends to have longer subword units than BPE. This is closer to the length distribution of gold-standard English morphs, which have a mean length of approximately 6 characters (Creutz and Linden, 2004).

Comparison with morphological segmenters In Table 3, we further corroborate these observations by performing a quantitative evaluation of the degree to which each unsupervised segmentation algorithm aligns with morphological baselines for each language. For English, we produce gold surface allomorph boundaries from the CELEX2 lexical database (Baayen et al., 1995) in the manner of Creutz and Lindén (2004). We then compare each algorithm's subword unit boundaries with gold morpheme boundaries for words with 2 or more morphemes, weighted by their frequency in English Wikipedia. For Japanese, we compare subword tokenizations of Japanese Wikipedia sentences to morphological reference tokenizations produced using the MeCab morphological analysis and tokenization tool (Kudo, 2006) using version 2.3.0 of the UniDic dictionary (Den et al., 2007).

We find that for both languages, the segmentations produced by the unigram LM method correspond more closely to the morphological references, confirming our qualitative analysis. On English data, both unsupervised methods exhibit low boundary recall; we attribute this to the fact that they represent many common words with underlying derivational morphology as single tokens, although for BPE this is compounded by effects we discuss in Section 3.2.

The ability of the unigram LM method to recover the morphological structure of the text without explicit supervision aligns with the main findings of Creutz and Lagus (2005), who successfully use maximum-a-posteriori unigram language models to perform unsupervised morphological segmentation of English and Finnish.

3.2 Vocabulary Allocation

By surfacing subword units that align with morphology, the unigram LM tokenization provides the opportunity for the model to learn composable subword embeddings. If an affix reliably signals a linguistic feature, rather than needing to store that information redundantly across the embeddings of many tokens containing the affix, the model can store it in just the embedding of the affix.

These results suggest that the unigram LM method may allocate its vocabulary more economically. We note in Figure 2b that both vocabularies contain a "dead zone" of tokens whose frequency is much lower than the rest of the vocabulary. This is largely the result of the presence of a number of very uncommon characters, including Chinese and Japanese kanji, in the training corpus. In the BPE tokenization, however, this effect is exacerbated, with the dead zone containing about 1500 more entries as a result of the tendency of its vocabulary construction process to produce intermediate "junk" tokens. For example, in the case where three tokens almost always occur as a group, in order to merge them into a single token, BPE must first merge one pair before incorporating the third token; this leaves an intermediate token in the vocabulary that will only occur rarely on its own. Additionally, tokens that appear in many contexts, such as inflectional affixes (-s, -ed), will tend to merge with many adjacent units due to their frequency. However, these merges lead to embedding redundancy, as these affixes usually have the same linguistic function in every context. Since the unigram LM method selects tokens during vocabulary construction using a global optimization procedure, it does not produce junk tokens; this property also allows it to avoid merging frequent tokens with their neighbors too aggressively.

Japanese vocabulary comparisons are included

[2]Note that the BPE vocabulary still includes these affixes, but when they are encountered during tokenization, they are almost always merged into larger units as in Figure 1.

Model	English SQuAD 1.1 (dev.) EM	English SQuAD 1.1 (dev.) F1	English MNLI (dev.) Acc. (m)	English MNLI (dev.) Acc. (mm)	English CoNLL NER Dev. F1	English CoNLL NER Test F1	Japanese TyDi QA (dev.) EM	Japanese TyDi QA (dev.) F1
Ours, BPE	$80.6 \pm .2$	$88.2 \pm .1$	$81.4 \pm .3$	$82.4 \pm .3$	$94.0 \pm .1$	$90.2 \pm .0$	41.4 ± 0.6	42.1 ± 0.6
Ours, Uni. LM	$81.8 \pm .2$	$89.3 \pm .1$	$82.8 \pm .2$	$82.9 \pm .2$	$94.3 \pm .1$	$90.4 \pm .1$	53.7 ± 1.3	54.4 ± 1.2
$\text{BERT}_{\text{BASE}}$	80.5	88.5	84.6	83.4	96.4	92.4	–	–

Table 4: Fine-tuning results. Metrics are averaged across 5 fine-tuning seeds with standard deviations indicated by $\pm$; due to computational constraints we did not pretrain more than once per tokenization. We include fine-tuning results for a transformer with a comparable architecture, $\text{BERT}_{\text{BASE}}$, for reference, although we note that a direct comparison cannot be made due to $\text{BERT}_{\text{BASE}}$ using both a larger pretraining corpus and a larger subword vocabulary.

in Appendix B.

4 Downstream Task Experiments

In order to make a fair experimental comparison between these two methods on downstream tasks, we do not use an existing pretrained language model like BERT, but instead train our own language models from scratch, controlling for the data, training objective, and optimization procedure. We pretrain four transformer masked language models using the architecture and training objective of RoBERTa-base (Liu et al., 2019) using the reference `fairseq` implementation (Ott et al., 2019). Two are pretrained on the text of English Wikipedia, comprising ∼3B tokens under either tokenization. The other two are pretrained on the text of Japanese Wikipedia, comprising ∼0.6B tokens. In each pair, one model is pretrained on the BPE tokenization of the corpus, and the other on the unigram LM tokenization, each with a vocabulary of 20,000 tokens. Hyperparameters are listed in Appendix A.

We subsequently fine-tune each of the pretrained English models on the SQuAD question-answering task (Rajpurkar et al., 2016), the MNLI textual entailment task (Williams et al., 2018), and the English portion of the CoNLL 2003 named-entity recognition shared task (Tjong Kim Sang and De Meulder, 2003). We fine-tune the Japanese models on the Japanese minimal-answer subset of the TyDi question-answering task (Clark et al., 2020). We base our fine-tuning implementations on those of the `transformers` toolkit (Wolf et al., 2019).

The results of our fine-tuning experiments are presented in Table 4. We show that fine-tuning models pretrained with unigram LM tokenization produces better performance than fine-tuning models pretrained with BPE tokenization for all tasks. These results suggest that the higher morphological plausibility of the unigram LM tokenization may translate into better downstream task performance as well. Larger performance gaps are evident on SQuAD and MNLI, but the largest gap appears on Japanese TyDi. Differences in pretraining may be more evident in this setting due to the fact that the Japanese portion of the TyDi training split only contains ∼5k examples, compared to the ∼88k examples available for fine-tuning on SQuAD. Additionally, written Japanese does not feature whitespace between words, so it is possible for tokenizations to differ in word boundary placement as well as subword segmentation.

5 Conclusion

In this work we show that the choice of input encoding makes a difference in how well pretrained language models are able to perform end tasks. This indicates that tokenization encodes a surprising amount of inductive bias, and we suggest that unigram LM tokenization may be the better choice for development of future pretrained models.

Acknowledgments

This work was partially supported by NSF Grant IIS-1814522 and a gift from Arm. This material is also based on research that is supported by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory (AFRL), DARPA, or the U.S. Government.

References

R. Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1995. The CELEX lexical database (release 2).

Jonathan Clark, Jennimaria Palomaki, Vitaly Nikolaev, Eunsol Choi, Dan Garrette, Michael Collins, and Tom Kwiatkowski. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8(0):454–470.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology Helsinki.

Mathias Creutz and Krister Lindén. 2004. Morpheme segmentation gold standards for finnish and english. Report, Helsinki University of Technology.

Mathias Johan Philip Creutz and Bo Krister Johan Linden. 2004. Morpheme segmentation gold standards for Finnish and English. *Publications in Computer and Information Science Report A77*.

Yasuharu Den, Toshinobu Ogiso, Hideki Ogura, Atsushi Yamada, Nobuaki Minematsu, Kiyotaka Uchimoto, and Hanae Koiso. 2007. The development of an electronic dictionary for morphological analysis and its application to japanese corpus linguistics. *Japanese Linguistics*, 22:101–123.

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.

Miguel Domingo, Mercedes Garcıa-Martınez, Alexandre Helle, and Francisco Casacuberta. 2018. How much does tokenization affect in neural machine translation? *arXiv preprint arXiv:1812.08621*.

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

Matthias Gallé. 2019. Investigating the effectiveness of BPE: The power of shorter sequences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1375–1381, Hong Kong, China. Association for Computational Linguistics.

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. 2006. MeCab: Yet another part-of-speech and morphological analyzer.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

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.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

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

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

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.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

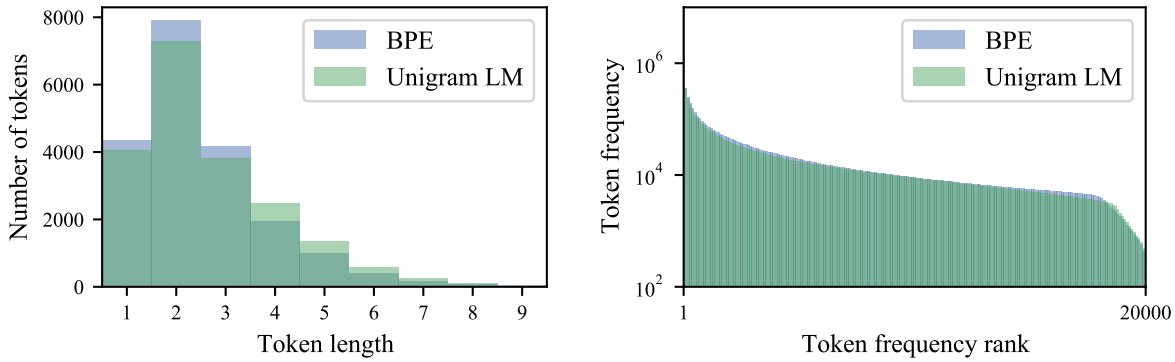
A Hyperparameters

Pretraining	
Model architecture	RoBERTa-base (Liu et al., 2019)
Implementation	`fairseq` (Ott et al., 2019)
Optimizer	Adam, $\epsilon =$ 1e-6 $\beta = (0.9, 0.98)$ (Kingma and Ba, 2015)
Learning rate decay	Polynomial
Peak learning rate	0.0005
Warmup steps	10000
Weight decay	0.01
Batch size	2048
Sequence length	512
Total updates	125000
MLP dropout	0.1
Attention dropout	0.1
Precision	16-bit
Fine-tuning	
Implementations	`transformers` (Wolf et al., 2019)
Optimizer	Adam, $\epsilon =$ 1e-8 $\beta = (0.9, 0.999)$
Learning rate decay	Linear
Peak learning rate	5e-5
Warmup steps	0
Weight decay	0
Batch size	32
Sequence length (SQuAD, TyDi QA)	512
Passage stride (SQuAD, TyDi QA)	192
Sequence length (MNLI, NER)	128
Epochs	3
Precision	16-bit
Tokenization	
Implementations	SentencePiece (Kudo and Richardson, 2018)
Vocabulary size	20000
Unigram LM α	0.25

B Japanese vocabulary comparison

More frequent in									
BPE					Unigram LM				
)、	)。	_)	ンの	スの	li	lo	ていく	vi	てしまう
は	のは	_の	、2	ンは	hi	0%	to	no	ta

Table 5: Tokens with the highest difference in frequency between tokenizations. The BPE method merges common tokens, such as particles and punctuation, even when they do not form meaningful units. The unigram LM method recovers the units ていく and てしまう, which are productive components of the Japanese verb conjugation system.



(a) Token length distributions within each vocabulary

(b) Token frequency profiles over the corpus

Figure 3: Japanese subword vocabulary and corpus profiles. (a) The unigram LM method produces longer tokens, as it does in English. (b) Token frequency profiles resemble those of English, though the effect of the "dead zone" is less pronounced.