An Embarrassingly Simple Method to Mitigate `und` `es` `ira` `ble` Properties of Pretrained Language Model Tokenizers

Valentin Hofmann[*‡], **Hinrich Schütze**[‡], **Janet B. Pierrehumbert**[†*]

[*]Faculty of Linguistics, University of Oxford
[†]Department of Engineering Science, University of Oxford
[‡]Center for Information and Language Processing, LMU Munich
valentin.hofmann@ling-phil.ox.ac.uk

Abstract

We introduce FLOTA (Few Longest Token Approximation), a simple yet effective method to improve the tokenization of pretrained language models (PLMs). FLOTA uses the vocabulary of a standard tokenizer but tries to preserve the morphological structure of words during tokenization. We evaluate FLOTA on morphological gold segmentations as well as a text classification task, using BERT, GPT-2, and XLNet as example PLMs. FLOTA leads to performance gains, makes inference more efficient, and enhances the robustness of PLMs with respect to whitespace noise.

1 Introduction

The first step in NLP architectures using pretrained language models (PLMs) is to map text to a sequence of tokens corresponding to input embeddings. The tokenizers used to accomplish this have been shown to exhibit various undesirable properties such as generating segmentations that blur word meaning (Bostrom and Durrett, 2020; Church, 2020; Hofmann et al., 2021) and generalizing suboptimally to new domains (Tan et al., 2020; Hong et al., 2021; Sachidananda et al., 2021).

In this paper, we propose **FLOTA** (**F**ew **Lo**ngest **T**oken **A**pproximation), a simple yet effective method to mitigate some shortcomings of PLM tokenizers. FLOTA is motivated by the following hypothesis: rather than finding a segmentation that *covers all characters* of a word but *destroys its morphological structure*, it can be more beneficial to find a segmentation that *does not cover all characters* but *preserves key aspects of the morphology*. We confirm this hypothesis in this paper.

Our study investigates three PLMs and corresponding tokenizers: BERT (base, uncased; Devlin et al., 2019), which uses WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016), GPT-2 (base, cased; Radford et al., 2019), which uses byte-pair encoding (BPE; Gage, 1994; Sennrich et al., 2016), and XLNet (base, cased; Yang et al., 2019), which uses Unigram (Kudo, 2018). We find that FLOTA increases the morphological quality of all tokenizers as evaluated on human-annotated gold segmentations as well as the performance of all PLMs on a text classification challenge set.

Contributions. We introduce FLOTA, a simple yet effective method to improve the tokenization of PLMs during finetuning. FLOTA uses the vocabulary of a standard tokenizer but tries to preserve the morphological structure of words during tokenization. We show that FLOTA has three advantages compared to standard tokenization: (i) it can increase the performance of PLMs on certain tasks, sometimes substantially; (ii) it makes inference more efficient by shortening the processed token sequences; (iii) it enhances the robustness of PLMs with respect to certain types of noise in the data. All this is achieved *without requiring any additional parameters or resources* compared to vanilla PLM finetuning. We also release a text classification challenge set that can serve as a benchmark for future studies on PLM tokenizers.[1]

2 Few Longest Token Approximation

Let V be a set of tokens that constitute the vocabulary of a tokenizer. For the tokenizers discussed in this paper, V contains words, subwords, and characters. Let ϕ be a model used by the tokenizer to map text to a sequence of tokens from V.

FLOTA (Few Longest Token Approximation) discards ϕ and uses V in a modified way. Given a word w not in V, FLOTA tokenizes it by determining the longest substring $s \in V$ of w, returning s, and recursing on $w \setminus s$, the string(s) remaining when s is removed from w. We stop after k recursive calls or when the residue is null. Figure 1 provides pseudocode. For the example word *undesirable* and $k = 2$, FLOTA first searches on

[1]We make our code and data available at https://github.com/valentinhofmann/flota.

```
MaxSubwordSplit($w, V$)
1  $l = length(w)$
2  for $j = l$ downto $0$
3      for $i = 0$ to $l - j + 1$
4          $s = w[i\,..\,i + j]$
5          if $s \in V$
6              $r = w[0\,..\,i] \oplus_j w[i + j\,..\,l]$
7              return $s, r, i$

FlotaTokenize($w, k, V$)
1  $s, r, i =$ MaxSubwordSplit($w, V$)
2  if $k == 1$ or $hyphen(r)$
3      $F = \{\}$
4      $F[i] = s$
5      return $F$
6  $F =$ FlotaTokenize($r, k - 1, V$)
7  $F[i] = s$
8  return $F$
```

Figure 1: FLOTA pseudocode. FLOTA is based on a recursive function FLOTATOKENIZE that uses a hash table F to store the longest substring s and its index i on each recursive call. s and i are found by means of a second function MAXSUBWORDSPLIT, which also returns a residue r. In practice, to ensure correct indexing throughout different recursive calls as well as prevent using discontinuous substrings for tokenization, we compute r using an operation $\oplus_j$ that concatenates two strings by putting j (length of s) hyphens between them. The recursion stops after k recursive calls or when r only consists of hyphens (determined by a boolean function $hyphen$). The hash table returned by FLOTATOKENIZE is converted to a tokenization using a simple wrapper function that sorts the found substrings by their indices (not shown). If MAXSUBWORDSPLIT does not find a substring $s \in V$, FLOTATOKENIZE returns an empty hash table (not shown).

undesirable and finds `desirable`, then searches on `un---------` and finds `un`, then stops (since $k = 2$; it would also stop for $k > 2$ since the residue is null) and returns the tokenization `un`, `desirable`. The WordPiece tokenization, on the other hand, is `und`, `es`, `ira`, `ble`.

FLOTA is guided by the following observations: many words not in V are made up of smaller and typically more frequent elements that determine their meaning (e.g., they are derivatives such as *undesirable*); many of these elements are in V.[2] By recursively searching for the longest substrings, we hope to recover the most important meaningful elements. This is also why it makes sense to stop after k recursions: if FLOTA returns the most important meaningful elements as the first few tokens, we expect to not lose much by stopping.

[2]Existing tokenizers have been shown to be able to recover these elements only to a very limited extent (Bostrom and Durrett, 2020; Church, 2020; Hofmann et al., 2021).

Model	Tokenization	$\overline{C}$	$\overline{R}$	$\overline{M}$
BERT	FIRST	.869	.817	.664
BERT	LONGEST	.865	.797	.664
BERT	FLOTA	**.990**	**.876**	**.896**
GPT-2	FIRST	.878	.674	.625
GPT-2	LONGEST	.874	.674	.625
GPT-2	FLOTA	**.988**	**.845**	**.861**
XLNet	FIRST	.886	.820	.724
XLNet	LONGEST	902	.845	.756
XLNet	FLOTA	**.992**	**.900**	**.922**

Table 1: Morphological quality. $\overline{C}$: morphological coverage ($k = 2$); $\overline{R}$: stem recall; $\overline{M}$: full match.

3 Evaluation on Gold Segmentations

English inflection is simple, but the language has highly complex word formation, i.e., derivation and compounding (Cotterell et al., 2017; Pierrehumbert and Granell, 2018). To evaluate the morphological quality of FLOTA against the standard tokenizers, we thus focus on derivatives and compounds.

Data. Our evaluation uses CELEX (Baayen et al., 1995) and LADEC (Gagné et al., 2019), two large datasets of human-annotated gold segmentations of morphologically complex words. We merge both datasets and extract all words consisting of a prefix and a stem (prefixed derivatives), a stem and a suffix (suffixed derivatives), or two stems (compounds). We create for each PLM a subset of words where both morphological elements (i.e., stems and affixes) are in the tokenizer vocabulary, but the word itself is not in the tokenizer vocabulary. In such cases, a word needs to be segmented, and it is guaranteed that *the gold segmentation is possible* given the tokenizer vocabulary. This procedure results in 11,272, 11,253, 10,848 words for BERT, GPT-2, XLNet, respectively.

Experimental Setup. We define three metrics to analyze how closely FLOTA matches the gold segmentations. We compare against two alternative tokenization strategies: representing words as the k first tokens returned by the standard tokenizer (FIRST) and representing words as the k longest tokens returned by the standard tokenizer (LONGEST). Recall that the WordPiece tokenization of the running example *undesirable* is `und`, `es`, `ira`, `ble`. With $k = 3$, FIRST is `und`, `es`, `ira` (i.e., it simply returns the first k tokens) and LONGEST is `und`, `ira`, `ble` (i.e., it returns the k

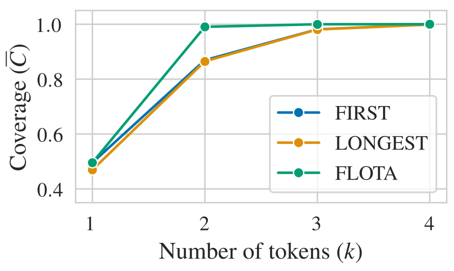

Figure 2: Morphological coverage for varying k.

longest tokens in the order in which they occur in the standard tokenization).

Morphological coverage. We analyze what proportion of morphological elements is covered by each tokenization strategy for varying k, a measure that we call *morphological coverage*, C. For *undesirable* and $k = 3$, FIRST and LONGEST contain *un* ($C = 0.5$) while FLOTA contains both *un* and *desirable* ($C = 1$). We compute the mean morphological coverage across all words, $\overline{C}$.

We find that for all three tokenizers, FLOTA already covers about 99% of the morphological elements with just $k = 2$, a value that FIRST and LONGEST only reach with $k = 4$ (Table 1, Figure 2), indicating that FLOTA needs considerably fewer tokens than the standard tokenization to convey the same amount of semantic and syntactic information. This can also be seen by examining the average number of tokens needed to fully tokenize a word (i.e., $k = \infty$), with the values for FLOTA (BERT: 2.02; GPT-2: 2.03; XLNet: 2.02) being lower than the values for the standard tokenization (BERT: 2.30; GPT-2: 2.23; XLNet: 2.26). The pairwise differences are statistically significant ($p < 0.001$) as shown by two-tailed t-tests.

Stem recall. Given its relevance for the overall lexical meaning of a word, we are interested in how often FLOTA returns the stem at $k = 1$. We test this using a measure that we call *stem recall*, R ($R = 1$ if the token is the stem,[3] otherwise $R = 0$), and compute the mean stem recall $\overline{R}$ across all words. We again compare with FIRST and LONGEST. Notice the stem according to the gold segmentation is longer than the second morphological element in 97% of the examined complex words, which means that LONGEST provides a close estimate of how often the full standard tokenization contains the stem (since any other element in the full standard tokenization is shorter and hence very unlikely to be the stem).

[3]For compounds: one of the two stems.

FLOTA returns the stem considerably more often than either FIRST or LONGEST, but there are clear differences between the models (Table 1): for GPT-2, FLOTA increases $\overline{R}$ by more than 15% while the difference amounts to 5% for XLNet.

Full match. Extending the evaluation of stem recall, we examine whether the tokenization at $k = 2$ is identical to the gold segmentation (which always has two elements) using a measure that we call *full match*, M ($M = 1$ if the tokenization exactly matches the gold segmentation, otherwise $M = 0$). We again compute the mean value $\overline{M}$ across all words. Here, the values for both FIRST and LONGEST are identical to the performance of the full standard tokenization: for the full standard tokenization to exactly match a segmentation of two elements, it must consist of two tokens, and hence it is necessarily equal to both its first two tokens and its longest two tokens.[4] Table 1 shows that FLOTA substantially improves $\overline{M}$.

The evaluation on gold segmentations indicates that FLOTA increases the morphological quality of PLM tokenizers compared to the standard tokenization and simple alternatives. We also find underlying differences in the morphological quality of the tokenizers, with BPE and Unigram lying at the negative and positive extremes, in line with prior work (Bostrom and Durrett, 2020). Our analysis shows that WordPiece lies in between.

4 Evaluation on Downstream Task

We investigate whether the enhanced quality of FLOTA tokenizations translates to performance on downstream tasks. We focus on text classification as one of the most common tasks in NLP.

Data. We create two text classification challenge sets based on ArXiv,[5] each consisting of three datasets. Specifically, for the subject areas of computer science, maths, and physics, we extract titles for the 20 most frequent subareas (e.g., *Computation and Language*). We then sample 100/1,000 titles per subarea, resulting in three text classification datasets of 2,000/20,000 titles each, which we bundle together as ArXiv-S/L. Our sampling

[4]Surprisingly, this does not hold for Unigram, which sometimes creates *separate* start-of-word tokens; e.g., the Unigram tokenization of *americanize* is _, `american`, `ize`, where _ is a start-of-word token. Notice that in such cases (with $k = 2$), LONGEST (`american`, `ize`) matches the gold segmentation while FIRST (_, `american`) does not, explaining the performance difference for XLNet.

[5]`kaggle.com/Cornell-University/arxiv`

	ArXiv-S		ArXiv-L	
Model	Dev	Test	Dev	Test
BERT	.469	.470	.674	.659
+FLOTA	**.491**	**.485**	**.675**	**.661**
GPT-2	.329	.324	.526	.507
+FLOTA	**.353**	**.382**	**.558**	**.542**
XLNet	.435	**.454**	.660	.641
+FLOTA	**.446**	.428	**.664**	**.646**

Table 2: Performance. FLOTA leads to gains in averaged F1, particularly for BERT and GPT-2. Performance breakdowns for the individual datasets forming ArXiv-S/L are provided in Appendix A.3.

ensures that ArXiv-S/L require challenging generalization from a small number of short training examples with highly complex language. See Appendix A.1 for more details.

Experimental Setup. We split the six datasets of ArXiv-S and ArXiv-L into 60% train, 20% dev, and 20% test. We then train the three PLMs with classification heads on the six train splits, once with the standard tokenizers and once with FLOTA. See Appendix A.2 for hyperparameters. For FLOTA, we treat k as an additional tunable hyperparameter. We use F1 as the evaluation metric.

Performance. The FLOTA models perform better than the models with standard tokenization, albeit to varying degrees for the three PLMs (Table 2). The difference is most pronounced for GPT-2, with FLOTA resulting in large performance gains of up to 5%. In addition, GPT-2 performs worse than the other two PLMs on all datasets, suggesting that BPE is generally not a good fit for complex language. BERT also clearly benefits from using FLOTA, particularly on ArXiv-S. Out of the three considered PLMs, XLNet obtains the smallest performance gain from using FLOTA, but it still benefits in the majority of cases.

The advantage of FLOTA mirrors the differences observed in the morphological analysis, indicating that *FLOTA helps close the morphological quality gap* between standard tokenizations and gold segmentations. Where the gap is large, gains due to FLOTA are large (GPT-2/BPE); where it is small, gains due to FLOTA are small (XLNet/Unigram). BERT/WordPiece again lies in between.

Impact of k. To test how the performance varies with k, we focus on BERT and compare the FLOTA models for $k \in \{1, 2, 3, 4\}$ with the two alternatives FIRST and LONGEST from Section 3. See Appendix A.4 for hyperparameters.

Figure 3 shows that FLOTA only drops slightly as we decrease k, with the minimum F1 at $k = 1$

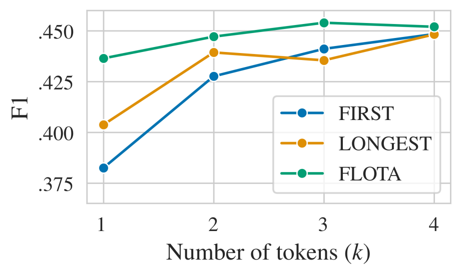

Figure 3: FLOTA is less impaired by smaller values of k (maximum number of tokens per word) than FIRST/LONGEST. Results are averaged F1 of BERT on ArXiv-S (dev/test merged).

		FLOTA			
Model	ST	$k=1$	$k=2$	$k=3$	$k=4$
BERT	12.9	8.3	10.5	11.4	11.6
GPT-2	12.9	8.3	10.7	11.5	11.8
XLNet	13.6	8.3	10.9	11.9	12.2

Table 3: Average sequence length of titles (ArXiv-L, physics). ST: standard tokenization.

(43.6%) lying less than 2% below the maximum F1 at $k = 3$ (45.4%). In contrast, FIRST and LONGEST drop substantially as we decrease k; for FIRST, the minimum F1 at $k = 1$ (38.2%) lies more than 6% below the maximum F1 at $k = 4$ (44.8%). The fact that FLOTA is more effective at preserving performance while reducing the number of tokens aligns with the observation that it covers a larger number of morphemes and hence more semantic and syntactic content than FIRST and LONGEST for small k (Section 3).

Efficiency. FLOTA allows to reduce the number of tokens used to tokenize text by varying k. Since the attention mechanism scales quadratically with sequence length (Peng et al., 2021), this has beneficial effects on the computational cost involved with employing a model trained using FLOTA. We empirically find that even for $k = 4$ (the largest value used in the experiments), token sequences generated by FLOTA are on average shorter than the token sequences generated by the standard tokenizations. Table 3 shows for one dataset (ArXiv-L, physics) the average sequence length of titles encoded with the standard tokenization versus FLOTA with varying $k \in \{1, 2, 3, 4\}$ for the three PLMs.

Robustness. To examine robustness against noise, a well-known problem for PLMs (Pruthi et al., 2019), we focus on missing whitespace between words (Soni et al., 2019). We randomly drop the whitespace between two adjacent words with

	ArXiv-S (N)		ArXiv-L (N)	
Model	Dev	Test	Dev	Test
BERT	.428	.412	.579	.554
+FLOTA	**.486**	**.447**	**.652**	**.632**
GPT-2	.313	.315	.481	.463
+FLOTA	**.359**	**.357**	**.541**	**.518**
XLNet	.392	.397	.609	.589
+FLOTA	**.434**	**.421**	**.641**	**.623**

Table 4: Performance with noise (N). FLOTA clearly increases F1 on ArXiv-S/L for all PLMs when input is noisy. See Appendix A.5 for hyperparameters. Performance breakdowns for the individual datasets forming ArXiv-S/L are provided in Appendix A.6.

probability $p = 0.3$ in ArXiv-S/L. We use *unseen* noise, i.e., we only inject noise during evaluation, not training, which is the more realistic and challenging scenario (Xue et al., 2021).

The results show that synthetic noise increases the performance gap between FLOTA and standard tokenization (Table 4). While there is a drop in performance for all models compared to the experiments without noise, the drop is much more pronounced for standard tokenization; e.g., BERT's performance on ArXiv-L (test) drops by 3% with FLOTA, but by 10% without it.

5 Limitations

While we find FLOTA to work well on text classification, there are tasks for which FLOTA might prove a less suitable tokenization method: e.g., for small values of k, FLOTA often discards suffixes, which can be important for tasks with a syntactic component such as POS tagging.

Similar considerations hold for transfer to languages other than English: e.g., in the case of languages with a non-linear morphology such as Arabic, FLOTA is expected to inherit the insufficiencies of the underlying tokenizer (Alkaoud and Syed, 2020; Antoun et al., 2020).

6 Related Work

The question how PLMs are affected by their tokenizer has attracted growing interest recently. Bostrom and Durrett (2020), Church (2020), Klein and Tsarfaty (2020), and Hofmann et al. (2021) focus on the linguistic properties of tokenizers. We contribute to this line of work by conducting the first comparative analysis of all three common PLM tokenizers and releasing a challenge set as a benchmark for future studies. Another strand of research has sought to improve PLM tokenizers by training models from scratch (Clark et al., 2021; Si et al., 2021; Xue et al., 2021; Zhang et al., 2021) or modifying the tokenizer during finetuning, mostly by adding tokens and corresponding embeddings (Chau et al., 2020; Tan et al., 2020; Hong et al., 2021; Sachidananda et al., 2021). FLOTA crucially differs in that it can be used during finetuning but does not add any parameters to the PLM. Furthermore, there has been work improving tokenization by variously exploiting the probabilistic nature of tokenizers (Kudo, 2018; Provilkov et al., 2020; Cao and Rimell, 2021). By contrast, our method does not need access to the underlying model.

Our study also relates to computational work on derivational morphology (Cotterell et al., 2017; Vylomova et al., 2017; Cotterell and Schütze, 2018; Deutsch et al., 2018; Hofmann et al., 2020a,b,c) and word segmentation (Cotterell et al., 2016; Kann et al., 2016; Ruzsics and Samardžić, 2017; Mager et al., 2019, 2020; Seker and Tsarfaty, 2020; Amrhein and Sennrich, 2021). We are the first to systematically evaluate the segmentations of PLM tokenizers on human-annotated gold data.

Conceptually, the findings of our study are in line with evidence from the cognitive sciences that knowledge of a longer (i.e., more detailed and informative) sequence takes priority over any knowledge about smaller sequences (Caramazza et al., 1988; Laudanna and Burani, 1995; Baayen et al., 1997; Needle and Pierrehumbert, 2018).

7 Conclusion

We introduce FLOTA (Few Longest Token Approximation), a simple yet effective method to improve the tokenization of pretrained language models (PLMs). FLOTA uses the vocabulary of a standard tokenizer but tries to preserve the morphological structure of words during tokenization. FLOTA leads to performance gains, makes inference more efficient, and substantially enhances the robustness of PLMs with respect to whitespace noise.

Acknowledgements

This work was funded by the European Research Council (#740516) and the Engineering and Physical Sciences Research Council (EP/T023333/1). The first author was also supported by the German Academic Scholarship Foundation and the Arts and Humanities Research Council. We thank the reviewers for their helpful comments.

Ethical Considerations

FLOTA shortens the average length of sequences processed by PLMs, thus reducing their energy requirements, a desirable property given their otherwise detrimental environmental footprint (Schwartz et al., 2019; Strubell et al., 2019).

References

Mohamed Alkaoud and Mairaj Syed. 2020. On the importance of tokenization in Arabic embedding models. In *Arabic Natural Language Processing Workshop (WANLP) 5*.

Chantal Amrhein and Rico Sennrich. 2021. How suitable are subword segmentation strategies for translating non-concatenative morphology? In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT)*.

R. Harald Baayen, Ton Dijkstra, and Robert Schreuder. 1997. Singulars and plurals in Dutch: Evidence for a parallel dual-route model. *Journal of Memory and Language*, 37:94–117.

R. Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1995. *The CELEX lexical database (CD-ROM)*. Linguistic Data Consortium, Philadelphia, PA.

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

Kris Cao and Laura Rimell. 2021. You should evaluate your language model on marginal likelihood over tokenisations. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2021*.

Alfonso Caramazza, Alessandro Laudanna, and Cristina Romani. 1988. Lexical access and inflectional morphology. *Cognition*, 28(297-332).

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020. Parsing with multilingual BERT, a small corpus, and a small treebank. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Kenneth Church. 2020. Emerging trends: Subwords, seriously? *Natural Language Engineering*, 26(3):375–382.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2021. CANINE: Pre-training an efficient tokenization-free encoder for language representation. In *arXiv 2103.06874*.

Ryan Cotterell and Hinrich Schütze. 2018. Joint semantic synthesis and morphological analysis of the derived word. *Transactions of the Association for Computational Linguistics*, 6:33–48.

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016. A joint model of orthography and morphological segmentation. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT) 2016*.

Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov, and David Yarowsky. 2017. Paradigm completion for derivational morphology. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2017*.

David Crystal. 1997. *The Cambridge encyclopedia of the English language*. Cambridge University Press, Cambridge, UK.

Daniel Deutsch, John Hewitt, and Dan Roth. 2018. A distributional and orthographic aggregation model for English derivational morphology. In *Annual Meeting of the Association for Computational Linguistics (ACL) 56*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HTL) 2019*.

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

Christina L. Gagné, Thomas L. Spalding, and Daniel Schmidtke. 2019. LADEC: The large database of English compounds. *Behavior Research Methods*, 51(5):2152–2179.

Valentin Hofmann, Janet B. Pierrehumbert, and Hinrich Schütze. 2020a. DagoBERT: Generating derivational morphology with a pretrained language model. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2020*.

Valentin Hofmann, Janet B. Pierrehumbert, and Hinrich Schütze. 2020b. Predicting the growth of morphological families from social and linguistic factors. In *Annual Meeting of the Association for Computational Linguistics (ACL) 58*.

Valentin Hofmann, Janet B. Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Improving BERT's interpretations of complex words with derivational morphology. In *Annual Meeting of the Association for Computational Linguistics (ACL) 59*.

Valentin Hofmann, Hinrich Schütze, and Janet B. Pierrehumbert. 2020c. A graph auto-encoder model of derivational morphology. In *Annual Meeting of* *the Association for Computational Linguistics (ACL) 58.*

Jimin Hong, Taehee Kim, Hyesu Lim, and Jaegul Choo. 2021. AVocaDo: Strategy for adapting vocabulary to downstream domain. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2021.*

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2016.*

Diederik P. Kingma and Jimmy L. Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR) 3.*

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

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Annual Meeting of the Association for Computational Linguistics (ACL) 56.*

Alessandro Laudanna and Cristina Burani. 1995. Distributional properties of derivational affixes: Implications for processing. In Laurie B. Feldman, editor, *Morphological aspects of language processing*, pages 345–364. Lawrence Erlbaum, Hillsdale, NJ.

Manuel Mager, Özlem Çetinoğlu, and Katharina Kann. 2019. Subword-level language identification for intra-word code-switching. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HTL) 2019.*

Manuel Mager, Özlem Çetinoğlu, and Katharina Kann. 2020. Tackling the low-resource challenge for canonical segmentation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2020.*

Jeremy M. Needle and Janet B. Pierrehumbert. 2018. Gendered associations of english morphology. *Journal of the Association for Laboratory Phonology*, 9(1):119.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. 2021. ABC: Attention with bounded-memory control. In *arXiv 2110.02488.*

Janet B. Pierrehumbert and Ramon Granell. 2018. On hapax legomena and morphological productivity. In *Workshop on Computational Research in Phonetics, Phonology, and Morphology (SIGMORPHON) 15.*

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Annual Meeting of the Association for Computational Linguistics (ACL) 58.*

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Annual Meeting of the Association for Computational Linguistics (ACL) 57.*

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

Tatyana Ruzsics and Tanja Samardžić. 2017. Neural sequence-to-sequence learning of internal word structure. In *Conference on Computational Natural Language Learning (CoNLL) 21.*

Vin Sachidananda, Jason S. Kessler, and Yi-An Lai. 2021. Efficient domain adaptation of language models via adaptive tokenization. In *Workshop on Simple and Efficient Natural Language Processing (SustaiNLP) 2.*

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 37.*

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. In *arXiv 1907.10597.*

Amit Seker and Reut Tsarfaty. 2020. A pointer network architecture for joint morphological segmentation and tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2020.*

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Annual Meeting of the Association for Computational Linguistics (ACL) 54.*

Chenglei Si, Zhengyan Zhang, Yingfa Chen, Fanchao Qi, Xiaozhi Wang, Zhiyuan Liu, and Maosong Sun. 2021. SHUOWEN-JIEZI: Linguistically informed tokenizers for Chinese language model pretraining. In *arXiv 2106.00400.*

Sandeep Soni, Lauren F. Klein, and Jacob Eisenstein. 2019. Correcting whitespace errors in digitized historical texts. In *Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature 3.*

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Annual Meeting of the Association for Computational Linguistics (ACL) 57.*

Samson Tan, Shafiq Joty, Lav R. Varshney, and Min-Yen Kan. 2020. Mind your inflections! Improving NLP for non-standard Englishes with base-inflection encoding. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2020.*

Ekaterina Vylomova, Ryan Cotterell, Timothy Baldwin, and Trevor Cohn. 2017. Context-aware prediction of derivational word-forms. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL) 15*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc Le V, 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. In *arXiv 1609.08144*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021. ByT5: Towards a token-free future with pre-trained byte-to-byte models. In *arXiv 2105.13626*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems (NeurIPS) 33*.

Xinsong Zhang, Pengshuai Li, and Hang Li. 2021. AMBERT: A pre-trained language model with multi-grained tokenization. In *Findings of the Association for Computational Linguistics: ACL 2021*.

A Appendix

A.1 Preprocessing

We exclude texts written in a language other than English and lowercase all words. We exclude titles with less than three and more than ten words. For each title, we compute the proportion of words starting with a productive prefix from the list provided by Crystal (1997). During sampling, we then weight titles by this proportion in order to make the language contained within the datasets as complex and challenging as possible.

A.2 Hyperparameters

The vocabulary size is 28,996 for BERT, 50,257 for GPT-2, and 32,000 for XLNet. The number of trainable parameters is 109,497,620 for BERT, 124,455,168 for GPT-2, and 117,324,308 for XLNet. The classification head for all three models uses softmax as the activation function.

We use a batch size of 64 and perform grid search for the number of epochs $n \in \{1, \dots, 20\}$ and the learning rate $l \in \{1 \times 10^{-5}, 3 \times 10^{-5}, 1 \times 10^{-4}\}$ (selection criterion: F1). We tune l on ArXiv-L (physics) and use the best configuration on all datasets. For the FLOTA models, we additionally tune $k \in \{1, 2, 3, 4\}$ (selection criterion: F1). Models are trained with categorical cross-entropy as the loss function and Adam (Kingma and Ba, 2015) as the optimizer. Experiments are performed on a GeForce GTX 1080 Ti GPU (11GB).

A.3 Performance

Table 5 provides breakdowns of the performance for the individual datasets forming ArXiv-S/L.

A.4 Hyperparameters

All hyperparameters are as for the main experiment (see Appendix A.2). For the learning rate, we use the best configuration from the main experiment. For FIRST and LONGEST, we tune $k \in \{1, 2, 3, 4\}$ (selection criterion: F1), identically to FLOTA in the main experiment.

A.5 Hyperparameters

All hyperparameters are as for the main experiment (see Appendix A.2). For the learning rate, we use the best configuration from the main experiment.

A.6 Performance

Table 6 provides breakdowns of the performance for the individual datasets forming ArXiv-S/L.

	ArXiv-S						ArXiv-L					
	Dev			Test			Dev			Test		
Model	CS	MATH	PHYS	CS	MATH	PHYS	CS	MATH	PHYS	CS	MATH	PHYS
BERT	.546	.358	.502	.498	.407	.504	.682	.660	.679	.649	.653	.675
+FLOTA	.546	.414	.514	.483	.404	.567	.677	.663	.686	.652	.658	.672
GPT-2	.354	.281	.353	.316	.261	.395	.493	.506	.578	.465	.498	.559
+FLOTA	.348	.313	.398	.370	.323	.454	.520	.549	.603	.498	.540	.587
XLNet	.473	.357	.476	.489	.358	.515	.654	.643	.684	.627	.642	.655
+FLOTA	.450	.402	.486	.415	.346	.522	.660	.651	.681	.633	.641	.665

Table 5: Performance (F1). CS: computer science; MATH: mathematics; PHYS: physics.

	ArXiv-S (N)						ArXiv-L (N)					
	Dev			Test			Dev			Test		
Model	CS	MATH	PHYS	CS	MATH	PHYS	CS	MATH	PHYS	CS	MATH	PHYS
BERT	.479	.333	.470	.566	.566	.605	.417	.338	.481	.531	.544	.588
+FLOTA	.548	.400	.511	.652	.640	.664	.452	.372	.518	.631	.620	.644
GPT-2	.336	.261	.342	.452	.461	.530	.326	.252	.366	.423	.454	.511
+FLOTA	.358	.316	.402	.514	.527	.582	.370	.296	.405	.481	.511	.562
XLNet	.431	.311	.433	.607	.594	.625	.470	.300	.421	.587	.576	.605
+FLOTA	.432	.398	.474	.646	.623	.655	.435	.360	.466	.627	.612	.631

Table 6: Performance (F1) with noise (N). CS: computer science; MATH: mathematics; PHYS: physics.