PatchBERT: Just-in-Time, Out-of-Vocabulary Patching

Sangwhan Moon
Tokyo Institute of Technology
Odd Concepts Inc.
Tokyo, Japan
sangwhan@iki.fi

Naoaki Okazaki
Tokyo Institute of Technology
Tokyo, Japan
okazaki@c.titech.ac.jp

Abstract

Large scale pre-trained language models have shown groundbreaking performance improvements for transfer learning in the domain of natural language processing. In our paper, we study a pre-trained multilingual BERT model and analyze the OOV rate on downstream tasks, how it introduces information loss, and as a side-effect, obstructs the potential of the underlying model. We then propose multiple approaches for mitigation and demonstrate that it improves performance with the same parameter count when combined with fine-tuning.

1 Background

The advent of large scale language models pre-trained with large unannotated corpora has shown significant advancements in the domain of natural language processing, especially by demonstrating their effectiveness through transfer learning for downstream tasks (Howard and Ruder, 2018; Devlin et al., 2019; Conneau and Lample, 2019; Radford et al.), analogous to ImageNet (Deng et al., 2009) pre-trained backbones in the domain of computer vision. In the domain of natural language processing, new methods have made it possible to use internet corpora as a nearly free source for increasing the amount of data at an unprecedented scale during pre-training.

Additionally, new tokenization methods such as Byte-Pair Encoding (BPE, Sennrich et al. (2016)), WordPiece (Wu et al., 2016), SentencePiece (Kudo and Richardson, 2018), which break the lexicons into smaller subwords, have shown to be effective when applied to alphabetic languages to reduce the size of the vocabulary while increasing the robustness against out-of-vocabulary (OOV) in downstream tasks. This is especially powerful when combined with transfer learning. However, these tokenizers still operate at Unicode character levels - contrary to the names suggesting byte-level (which would completely mitigate OOV, as studied in Gillick et al. (2016)). Hence, the minimum size of the vocabulary is twice the size of all unique characters in the corpus, as subword tokenizers store each character in prefix and suffix form in the vocabulary. As OOV was a much more prevalent problem in the context of lexicon-based methods, there have been many methods, such as dictionary-based postprocessing (Luong et al., 2015) and distributional representation based substitution (Kolachina et al., 2017). Recently this has not been as actively studied in the context of subword tokenization as Latin languages are no longer affected.

For these reasons, when trained against a diverse set of languages, the vocabulary size increases proportionally to the number of languages supported. Existing models have sampled portions of entire corpora or relaxed constraints on character level coverage for these languages, to prevent the vocabulary from growing to an unmanageable scale. As of today, this is an unavoidable trade-off when training multilingual models. This introduces a bottleneck for downstream tasks since any character omitted causes information loss.

Training models for each language is the most straightforward possible mitigation. However, the downside is the cost for pre-training; acquiring a large corpus is a daunting task, and training a large model for many researchers can be financially infeasible. The high upfront cost leaves transfer learning on an open, multilingual model as an economically attractive alternative. Unfortunately, due to corpus imbalance during pre-training, minor languages, especially those with a diverse character set (such as CJK languages), OOV is likely to surface. Our motivation is to improve performance for these languages, without significantly increas-

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 7846–7852,
November 16–20, 2020.

Language	Example
INEWS (Chinese)	湖密山友 ——哒哒香花海之旅！ 湖密山友 [UNK] 香花海之旅！
Twitter (Japanese)	... １ ５回は押した💢💢💢💢💢... ... １ ５ 回 は 押 [UNK] ...
NSMC (Korean)	**재밌습니다.재밌습니다**. [UNK] . [UNK] .

Table 1: Examples of OOV in the task datasets.

ing the computation cost when using open-source pre-trained models.

In our work, we propose multiple approaches to mitigate OOV during fine-tuning. We compare each approach with no OOV mitigation, along with increasing vocabulary size as a secondary baseline.

2 Approach

The multilingual BERT model **bert-base-multilingual-cased** (Devlin et al., 2019) we used performs two-phase tokenization, first with whitespace followed by WordPiece. The tokenizer was modified to support a secondary vocabulary which points new words to existing words for our experiments, connected to a Transformers library (Wolf et al., 2019) BERT model. The approach consists of three steps.

First, we perform a complete corpus analysis and search for all OOV surfaces by tokenizing the task corpus. An OOV surface in the context of BERT is an entire space tokenized token. Whenever OOV occurs, we keep a record of the entire OOV surface, along with the context.

For each OOV surface, we brute-force search to find the maximally specific OOV subword surface. An OOV subword surface is an actual subword missing in an OOV surface. In this step, we compute a frequency table for both OOV and in-vocabulary subwords for a preference mechanism in the mitigation strategy. We observed that most cases of the OOV subword surface were caused by one character missing in the vocabulary during our experiments, which is a result of incomplete character coverage from the corpora used for pre-training.

Finally, we use this information to build a mitigation strategy for the OOV subwords. Here, we evaluate different algorithms for OOV mitigation, each of which we discuss in the individual method sections below. After applying OOV mitigation, we then optionally perform fine-tuning and evaluate against the baseline.

Substitution to mitigate OOV has been studied in (Kolachina et al., 2017). This method depends on part-of-speech tagging or a secondary corpus and model for similarity computation, which is challenging to apply in a subword model. The significance of our approach is that it works for subword models and it's practical applicability, as only a downstream task corpus and a pre-trained model is required.

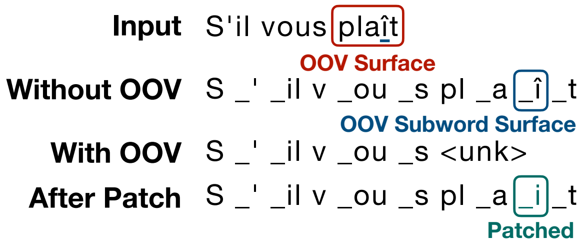

Figure 1: The hierarchy of OOV and the high level process explained through a simplified example in French. In this example, we assume î is a missing subword.

2.1 Surrogated Tokens

Surrogates, simply put, is treating a subword missing from the vocabulary to a subword that is already in the vocabulary of a pre-trained model. There are intuitive ways to find substitute words in a word-level setup, the most obvious being choosing a semantically similar word from a thesaurus. In a subword context, this is not as straightforward, as a subword generally has no meaning. In our work, we discuss different surrogate selection processes. The surrogate selection process introduces polysemy as a tradeoff for mitigating OOV. While some of the proposed methods add complexity for generative tasks, it does not increase the model's computation cost as the vocabulary size does not change.

The embeddings between the newly added subword and the surrogate are shared and updated together in the fine-tuning process. The OOV subword frequency table we constructed in the second step of the process above is used to break ties and minimize conflicts. For example, token A and B, both of which are OOV subwords, can end up with the same proposals $\{X, Y\}$ in preference order. In this case, given A has a higher frequency, it gets precedence over B, so the surrogate map becomes $A \rightarrow X$ and $B \rightarrow Y$. Our goal is to refine the proposals to be in a state where one surrogate is assigned to only one OOV token.

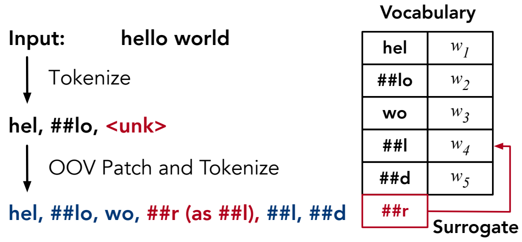

Figure 2: Surrogate vocabulary injection process.

6C10	汐	汑	汒	汓	汔	汕	汖	汗	汘	汙	汚	汛	汜	汝	汞	江
AC10	감	갑	값	갓	갔	강	갖	갗	갘	같	갚	갛	개	객	갞	갟

Figure 3: In character distance, The highlighted character is missing from the vocabulary. Observing the adjacent characters, in CJK ideographs they share a radical, while in Korean they share two subcharacters.

2.1.1 Character Distance

This method selects the surrogate with Unicode codepoint distance, limited to subword tokens within the vocabulary of the same length. In this process, we perform an exhaustive search, formulated as $\underset{w \in W'}{\mathrm{argmin}}(|\mathrm{ord}(v) - \mathrm{ord}(w)|_1$ where v is the OOV subword, and W' is a subset of the vocabulary W which satisfies UTF-8 length equality $|v| = |w|$ for $w \in W$. ord is Unicode ordinal conversion.

The intuition of this method builds on the characteristics of the CJK Unicode blocks, which allow us to cheaply approximate text or semantic similarity through the scalar values of the Unicode codepoints. The properties which we intend to exploit are different depending on the target language. In CJK ideographs, adjacent characters tend to share a radical, hence has a bias towards semantic similarity.

On the other hand, in Korean, phonetically similar characters are adjacent. This approximates edit distance, as a Korean character in Unicode is a combination of multiple sub-characters. This phonetic similarity differs from edit distance, as it tends to disallow edits on the first two components of the character. In the event of a distance tie, we used the candidate with a lower codepoint.

Frequent subword tokens get preferential treatment and hence get surrogates with closer distance to an infrequent token. Once a token has been assigned, it is not re-used as a surrogate.

2.1.2 Unseen Tokens

We select tokens from the in-vocabulary token frequency table, which were never seen in the current task as surrogates. As downstream tasks for evaluation do not require the entire vocabulary, we select random tokens with a frequency of 0 as surrogates.

This method is analogous to increasing the model parameters (via vocabulary size), then pruning back to the original size, but as an in-place operation. Any word previously assigned was held out to prevent re-assignment. As the vocabulary will have a large number of tokens never seen in most downstream tasks, we do not use any frequency preference here.

This method can also be combined with character distance to prefer surrogates with a high distance to assign surrogates in a distant Unicode page to prevent unseen tokens of the same language being used as surrogates.

2.1.3 Masked Language Model

First, the masked language model based proposal uses BERT's masked language head to generate surrogate proposals. Each subword OOV surface is replaced with the mask token and passed to the masked LM head with the whole context. The subword token with the highest probability is selected for each context, stored in a frequency table, to select the most common token later. We use the same frequency preference as character distance, which allows frequent OOV subwords to have precedence when selecting surrogates. As with other methods, once a surrogate is assigned, it is held out. Therefore, less frequent words are assigned to the next most locally frequent surrogate. After the entire process is done, OOV subwords that were not assigned a surrogate are assigned to the candidate with the lowest frequency.

2.2 Additional Tokens

Here, we add new tokens to the vocabulary and increase the model size, motivated by Wang et al. (2019). As this increases the network parameters, these are used as a secondary baseline to be compared with surrogates.

2.2.1 Random Initialization

After adding the missing subword to the vocabulary, the embedding is then randomly initialized.

2.2.2 Transfer Initialization

Transfer initialization is done by following the first step of the masked language model task to generate a list of surrogates. We then initialize by

Dataset	O/Tok	O/Sen	Total	%
NSMC	81603	60151	200K	30.1
KorQuAD	14159	8569	144K	5.9
Twitter	10310	5518	22K	25.1
INEWS	2570	1278	6K	20.1

Table 2: OOV analysis on the four datasets. O/Tok is the number of OOV tokens, O/Sen is the number of sentences with at least one OOV token, and the total sentence count, followed by the ratio of OOV sentences.

copying the embedding vector of the topmost probable candidate of the OOV subword into the newly added OOV subword's slot in the embedding matrix. These two tokens share the same initial embeddings but are expected to diverge through fine-tuning.

3 Datasets

For our experiments, we used four datasets for evaluation. For all tasks, we first learn OOV words, perform fine-tuning, then evaluate. The OOV rates noted for each dataset is the ratio of sentences containing at least one OOV token.

3.1 Naver Sentiment Movie Corpus

The Naver Sentiment Movie Corpus (NSMC) is a Korean sentiment analysis task, containing 200,000 user comments and a corresponding binary label which indicates positive or negative sentiment. The OOV rate was 30.1% due to a large number of typos and also being from a different domain.

3.2 Japanese Twitter Sentiment Analysis

As a second validation target language, we used a Japanese Twitter dataset, which is a sentiment analysis task with five possible labels. The task is 20K Tweets and 2K Tweets, respectively, for training and test. During analysis, we observed that a large portion of the OOV was from emojis, resulting in an OOV rate of 25.1%.

3.3 Chinese News Sentiment Analysis

The INEWS dataset is part of the ChineseGLUE dataset. The input is a short sentence from a news article, and the label is the tone of the article. This is also a sentiment analysis task, with a split of 5K train and 1K validation, and an OOV rate of 20.1%.

3.4 KorQuAD 1.0

KorQuAD 1.0 is a Korean version of the SQuAD (Rajpurkar et al., 2016) reading comprehension task. The task involves answering a question given a passage of text, and consists of 10K passages with 66K questions. The passages are from Wikipedia, which is commonly used as a part of large-scale training corpora. The result of this is a low OOV rate of 5.9%. For this task, task corpus fine-tuning was omitted to prevent the model from memorizing answers.

4 Results

The evaluation was done through the SST-2 GLUE task metrics (Wang et al., 2018) for the sentiment analysis tasks, and EM/F1 evaluation from the SQuAD metrics for KorQuAD, as the two tasks are compatible. Each model used the same dataset and training parameters as the baseline, only with different OOV mitigation methods.

Additionally, while Chinese and Japanese are both scriptio continua languages, BERT's tokenizer treats CJK ideograph text differently and breaks at every character. This makes the affected surface from OOV significantly smaller, resulting in less information loss. For these reasons, we expect to see larger gains in Korean, as the per-character break is not enabled.

4.1 Naver Sentiment Movie Corpus

Due to the larger OOV surface and frequency, we expect to observe a modest increase in the best case compared to the baseline. We can indeed observe that regardless of the mitigation method, OOV mitigation, in general, improves accuracy. The OOV tokens we observed here were from casual writing in user comments, which shifts from the book corpus like domain used for pre-train. This suggests that even without robust, representative embeddings, it is still better than losing information during tokenization. We also hypothesize that because the embeddings initially are not representative of the subword in context, performance improves by domain adaptation through fine-tuning.

4.2 Japanese Twitter Sentiment Analysis

This corpus showed a high OOV rate due to the frequent occurrence of emoji in the text. We observe similar patterns with the results from NSMC. Generally, we see improvements when both OOV mitigation and fine-tuning were done, except for character distance. We observed that character distance assigned surrogates to Korean characters, which may have contributed to this.

Model	Params+?	NSMC (ko)		Twitter (ja)		INEWS (zh)		KorQuAD (ko)	
		Acc@FT	Acc@NT	Acc@FT	Acc@NT	Acc@FT	Acc@NT	EM	F1
BERT (Baseline)	No	0.8773	0.8774	0.7348	0.7383	0.818	0.813	0.7012	0.8982
Add (Transfer)	Yes	0.8868	0.8812	0.7459	**0.7434**	0.820	0.810	0.7084	0.9022
Add (Random)	Yes	0.8882	0.8821	0.7449	0.7344	0.818	0.820	0.7085	0.9031
Char. Distance	No	**0.8885**	**0.8839**	0.7329	0.7394	**0.824**	0.818	**0.7101**	**0.9051**
Unseen Tokens	No	0.8876	0.8828	0.7354	0.7399	0.820	**0.828**	0.7021	0.9014
Masked LM	No	0.8853	0.8790	**0.7524**	0.7394	0.810	0.813	0.7064	0.9027
Best / Baseline Diff.		**0.0112**	**0.0065**	**0.0176**	**0.0051**	**0.006**	**0.015**	**0.0089**	**0.0069**

Table 3: Results. Acc denotes accuracy. Params denote a parameter increase. FT and NT mean with and without fine-tuning, respectively. Results for KorQuAD are without fine-tuning.

4.3 Chinese News Sentiment Analysis

While we observed a high OOV rate in this dataset, the improvement was negligible. Analyzing the surrogates, we observed that most of the OOV tokens were punctuation or uncommon ideographs, which we expected to, and confirmed to have little effect in the downstream task performance. In particular, we attribute the negligible gains to the nature of the task itself, as it is a news article classification task. While punctuation is an important aspect in tasks such as sentiment classification, classifying articles into categories has a stronger dependency on keywords, which are likely to in-vocabulary.

4.4 KorQuAD 1.0

We did not expect significant improvements due to the low OOV rate, and the results reflect this. While we still saw minor improvements across the board, the difference is incremental at best. The small delta can most likely be attributed to the relatively low OOV rate and omission of fine-tuning.

5 Conclusions

After demonstrating examples (1) and the effects of OOV triggered information loss, we propose multiple methods for mitigating OOV during downstream task fine-tuning. We then demonstrate and compare with no mitigation, mitigation through network modification, and surrogates, which require no network modification, and show how each approach affects downstream tasks. In particular, we show that vocabulary surrogates can provide performance boosts with no additional computation cost, especially when paired with fine-tuning.

We also empirically show that tasks with lower OOV suffer less when compared to languages that do not, as seen in table 1. While our experiments are limited to CJK languages on BERT, we believe the methods proposed are generic and simple to implement and expect the performance gains to also apply to different languages and models.

Acknowledgements

This paper is based on results obtained from a project, JPNP18002, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32*, pages 7057–7067. Curran Associates, Inc.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

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.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Prasanth Kolachina, Martin Riedl, and Chris Biemann. 2017. Replacing OOV words for dependency parsing with distributional semantics. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 11–19, Gothenburg, Sweden. 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 *EMNLP 2018 - Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Proceedings*.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. Technical report.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuad: 100,000+ questions for machine comprehension of text. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, and Dong Yu. 2019. Improving pre-trained multilingual model with vocabulary expansion. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 316–327, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

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, Lukasz 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. *CoRR*, abs/1609.08144.

Dataset	Parameter	Val
Fine-tune	Optimizer	Adam
	Adam ϵ	1e-8
	LR	5e-5
	GradAccum	1
	Weight Decay	0.0
	Batch Size	6
	Epochs	3
GLUE Tasks	Optimizer	Adam
	Adam ϵ	1e-8
	LR	2e-5
	GradAccum	1
	Weight Decay	0.0
	Batch Size	10
	Epochs	3
KorQuAD-SQuAD	Optimizer	Adam
	Adam ϵ	1e-8
	LR	3e-5
	GradAccum	1
	Weight Decay	0.0
	Batch Size	12
	Epochs	3

Table 4: Hyperparameters used to train each of the downstream task models.

A Appendix

A.1 Hyperparameters

We ran our experiments as close as possible to the baseline parameters used by the publicly available benchmark scripts for each task type. This means most of the hyperparameters for all of the evaluation was done as close to the default values as possible. The maximum sequence length was fixed to 512 for all models and tasks.

A.2 Environment

All experiments were executed on a shared rt_G.small instance on the ABCI compute cluster[1]. An rt_G.small node has 6 segregated CPU cores from a Xeon Gold 6148, a Tesla V100 GPU with 16GB VRAM, and 60GBs of memory. The training data and experimental code was streamed from a shared GPFS mount. Each experiment requires a

[1] https://abci.ai/

different amount of compute budget. The longest running experiment finished in 10 hours of wall clock time and the shortest finished in 2 hours of wall clock time. The average runtime for each experiment was approximately 5.5 hours.