Electronic Preprint for Journal of Information Processing Vol.29

Regular Paper

Effects and Mitigation of Out-of-vocabulary in Universal Language Models

Sangwhan Moon[1,†1,a)] Naoaki Okazaki[1,b)]

Received: December 8, 2020, Accepted: April 2, 2021

Abstract: One of the most important recent natural language processing (NLP) trends is transfer learning – using representations from language models implemented through a neural network to perform other tasks. While transfer learning is a promising and robust method, downstream task performance in transfer learning depends on the robustness of the backbone model's vocabulary, which in turn represents both the positive and negative characteristics of the corpus used to train it. With subword tokenization, out-of-vocabulary (OOV) is generally assumed to be a solved problem. Still, in languages with a large alphabet such as Chinese, Japanese, and Korean (CJK), this assumption does not hold. In our work, we demonstrate the adverse effects of OOV in the context of transfer learning in CJK languages, then propose a novel approach to maximize the utility of a pre-trained model suffering from OOV. Additionally, we further investigate the correlation of OOV to task performance and explore if and how mitigation can salvage a model with high OOV.

Keywords: Natural language processing, Machine learning, Transfer learning, Language models

1. Introduction

Using a large-scale neural language model as a pre-trained backbone and transferring to a multitude of downstream tasks [6], [7], [21] has been one of the most significant advancements in the field of natural language processing (NLP). This approach has been commonly used with convolutional neural networks trained against the ImageNet dataset and is commonly referred to as transfer learning. Language models used in this form do not yet have an official name but have been canonically called universal language models [11], due to its universal applicability. Unlike the domain of images or audio, pre-training language models for natural language processing do not require any annotated data due to various self-supervising training methods which have been recently proposed. This allows models to be pre-trained at scale, as there is a nearly infinite supply of training data from text data on the internet and through centuries worth of book corpora, given that one can efficiently digitize this into textual data and afford the amount of compute power needed as the training data is scaled up.

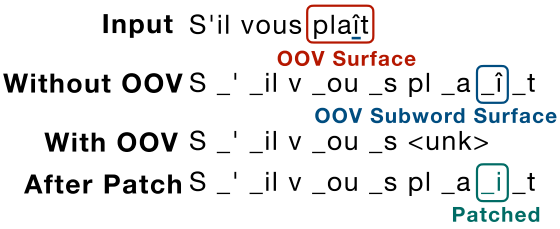
However, these methods still depend on a vocabulary bound to an embedding matrix. Due to the unavoidable growth of computational budget required as the vocabulary size increases, many methods have been proposed to reduce the vocabulary size to a manageable size, notably through subword based methods. Subword based methods, such as Byte-Pair Encoding (BPE) [23], WordPiece [30], SentencePiece [13], which break the lexicons into smaller subwords, have shown to be effective when applied to languages that utilize Latin-like alphabets in their writing system 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.

As these tokenizers still operate at Unicode character levels – contrary to the names suggesting byte-level (which would completely mitigate OOV, as studied in Ref. [10]). Hence, the vocabulary's minimum size 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. In commonly investigated languages, this still provides significantly more flexibility over lexicons. For these reasons, OOV issues have not been actively studied as it simply does not surface. However, this problem is yet to be solved in Chinese, Japanese, and Korean (CJK) languages due to the complex alphabets. We recognize that these unsolved problems make the applicability of neural language models for tasks in the context of CJK languages less universal than that of other languages.

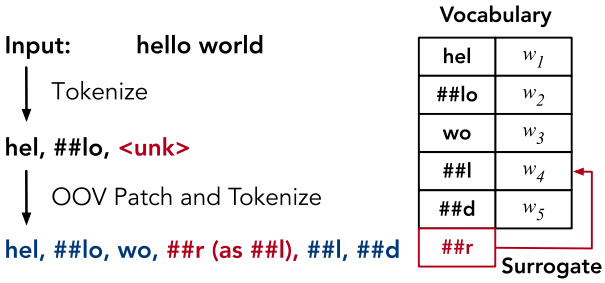
The high-level idea of our method is illustrated in **Fig. 1**, where î, a token missing from the vocabulary, is substituted with i. This work expands on our our existing work [20], which adds an OOV vocabulary learning step before fine-tuning for a downstream task. This is done by re-assigning OOV subwords to existing subwords. Through experiments, we demonstrate the effects of OOV in a downstream task setup and compare the OOV mitigation scheme's efficacy with and without additional pre-training. We further investigate how OOV contributes to task contribution by artificially inducing OOV in a pre-trained model and verify our proposed method can recover the model to a useable state even in moderately extreme OOV conditions.

[1] Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
[†1] Presently with Odd Concepts Inc., Meguro, Tokyo 153–0063, Japan
[a)] sangwhan@iki.fi
[b)] okazaki@c.titech.ac.jp

This work is an extension of our work in the Proceedings of Empirical Methods in Natural Language Processing, 2020 [20].



(a) Example of OOV and mitigation using a surrogate. Here, ## is expressed with _ to conserve space.



(b) Surrogate vocabulary data model explained. The surrogate vocabulary is treated as a mapping to an existing token.

Fig. 1 Overview of the vocabulary patching process explained with OOV examples.

2. Related Work

This work builds on the foundation of multiple natural language processing developments, which we will explain in the sections below. First, we disambiguate the concept of a language model, subword tokenization, then explain how it was combined to build a backbone model for different tasks and discuss the limitations.

2.1 Neural Language Models

A language model is formally defined as a probability distribution over a sequence of tokens. Given a sequence of length m, a probability $P(w_1, \ldots, w_m)$ is assigned to the whole sequence. A neural language model is a sequence prediction model, commonly implemented as a classifier that predicts a sequence based on the past context. In the example above, given a current time step t which satisfies $t < m$, it can predict $P(w_t|w_1, \ldots, w_{t-1})$ up to the $P(w_m|w_1, \ldots, w_{m-1})$. This is done by training such a neural network with unlabeled text corpora. This is possible because natural language can use coherently written sentences as a form of labeled data, unlike many other modalities. In the past, neural language models have been implemented using recurrent neural networks (RNN) such as long short-term memory networks (LSTM), but recent trends have shifted towards Transformer [26] based networks due to the discovery of its applicability in other tasks, which we will discuss in Section 2.3.

2.2 Subword Tokenization

Tokenization is the process of breaking a sequence into smaller units for an algorithm to consume. In an NLP context, the most common practice is to tokenize at the whitespace level, which generally results in tokens becoming words in languages such as English. However, when the algorithm that is expected to consume this is a neural network, the possible conjugations and permutations (such as numbers) become computationally intractable for the network to process. For these reasons, traditional methods used forms of preprocessing, such as stemming or lemmatization or sample the vocabulary to a consumable size. This results in information loss, which can result in lost performance during evaluation.

Additionally, these methods are not robust to rare words, which in evaluation can result in out-of-vocabulary. Subword tokenization was proposed as a mitigation for the out-of-vocabulary problem. Byte-Pair Encoding (BPE) [23], WordPiece [30], and SentencePiece [14] are all subword tokenization methods, which break the lexicons into smaller subwords. These methods have been shown to make the vocabulary robust to rare words while minimizing information loss.

2.3 Universal Language Models

The term Universal Language Model was first coined in ULMFiT [11], which uses a pre-trained LSTM language model's [18] hidden state to perform a multitude of tasks and achieve significant performance gains over per-task trained models. Around the same time, it was also discovered that such language model pre-training applied to a Transformer [26] based model, which was originally proposed for machine translation. The main difference of this Transformer based language model, BERT [7], is that it used significantly more compute power, was trained with a much larger pre-train dataset, and used subword tokenization.

With the advent of BERT, numerous other Transformer based architectures have been proposed and have shown to be extremely effective at being scaled up in terms of both increasing the amount of training data while also increasing the model capacity. Recent work such as GPT-3 [2] demonstrated substantial performance gains by scaling up both the model and data and validated that larger models are competitive in both zero-shot and few-shot settings.

Generally, the amount of pre-train data and model capacity is inversely proportional to the amount of downstream task data needed [2]. A pre-trained model also acts as a better initialization [1] as it also converges faster, reducing the amount of computation budget needed to achieve high performance on a given task. Due to this, state-of-the-art research employs transfer learning in some form.

While not all of the proposed backbone networks provide multilingual models or evaluations, work such as Ref. [6] shows that pre-training these models with multilingual corpora transferring with language models is also effective in a multilingual setup. However, a multilingual model's downside over a monolingual model is that multilingual models tend to suffer from pre-train data imbalance. This reflects the real world since different languages differ in the number of users the language has. Generally, the amount of textual data one can acquire tends to be proportional to the number of users of each language and is also affected by socio-economical factors contributing to a lower deployment

rate of digital technology, resulting in fewer data.

3. Preliminaries

3.1 CJK Tokenization in Universal Language Models

Out-of-vocabulary (OOV) in a subword tokenization context typically happens when a character was never seen in a pre-train context. These missing words can be introduced either by setting an upper limit on the character coverage or the vocabulary size. Here, the latter is tied to the former – if the character itself has been pruned from the initial set of characters to be covered, it is naturally infeasible to form a subword since the character is missing. This issue tends to be more significant in languages with a diverse alphabet, such as Chinese, Japanese, or Korean.

Chinese and Japanese use a large set of ideographs; some of which are rarely used, hence will not be statistically significant enough to be selected for inclusion but can be crucial in specific tasks such as named entity recognition (NER) since rare ideographs have moderate usage in names of people or locations.

Korean, on the other hand, has a large alphabet for somewhat artificial reasons. While the modern Korean alphabet can be expressed with 41 characters, the encoding of Korean in a computational context is done in a way that it is expressed through a combination of the underlying alphabet to form a composite character, and has been standardized in Unicode as these composite characters instead of its native form.

For these reasons, when trained against a diverse set of languages, the vocabulary size increases proportionally to the number of languages supported and needs to factor in the number of characters needed to express the language. For example, expressing English requires 26 characters, which results in 52 with both upper and lower cases. To express this in character level subwords, this results in 104 subwords, as both prefix and suffix forms are required in a vocabulary. To express French using the same vocabulary, the initial 26 can be re-used, and only 16 new characters specific to French are needed.

However, in the context of CJK languages, the initial character count is much larger, starting at approximately 2000 [*1] for common characters. Full coverage for the CJK ideographs requires 92,856 characters and 11,172 characters. While doubling the budget is not necessary, there are no cases in CJK languages [*2], the magnitude of budget required for the vocabulary is different from that of a language using a Latin alphabet.

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. The effect amplifies when a large character level vocabulary and scriptio continua languages [*3] exist in the same context, which is the case for all CJK languages. Examples of OOV in CJK languages can be seen in **Table 1**. Some methods have been proposed to mitigate this by decomposition of the characters [19], [24] to significantly reduce the vocabulary budget while retaining all information, but have shown little adoption in the wild.

Table 1 Examples of OOV in the task datasets. Here, we can observe that Chinese has OOV in punctuation, Japanese in emoji, and Korean in spelling errors.

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

In a monolingual setup, one can use pre-trained models for the target task language. However, when considering a multilingual setup, there is an additional layer of complexity by using an ensemble of monolingual models, as language detection is required to determine which model and tokenization scheme to use for each input. The most straightforward approach here would be to pre-train a monolingual model with a shared tokenization scheme for all the required languages.

However, the downside is the cost for pre-training; acquiring a large corpus is a daunting task, and training a large multilingual model for many researchers can be financially infeasible. The high upfront cost and complexity when implementing a multilingual system leaves transfer learning on an open, multilingual model as an economically attractive alternative. Unfortunately, due to corpus imbalance during pre-training, less-investigated languages, especially those with a diverse character set (such as CJK languages), OOV is likely to surface. Our motivation is to improve these languages' performance without significantly increasing the computation cost when using open-source pre-trained models.

3.2 BERT Tokenizer

The multilingual BERT model **bert-base-multilingual-cased** [7] we used performs two-phase tokenization, first with whitespace (token) followed by WordPiece [30] tokenization (subword token). An example output of the tokenizer is explained in Fig. 1. The prefix forms of the subwords are expressed in their original form, while suffix forms are expressed by appending a ## prefix.

If either form of the subword is missing in a token, the tokenization fails, and the token surface is treated as OOV. Using the example in Fig. 1, the suffix form of **î** is not in the vocabulary, hence the entire surface of the token **plaît** becomes OOV. This is due to the greedy merging nature of the WordPiece algorithm and is not universal to all subword-based methods.

Due to the dependency on initial whitespace tokenization, BERT's tokenization is not expected to work well with scriptio continua languages, especially if it has a diverse alphabet. To workaround this limitation, BERT's tokenizer implements special

[*1] This approximation is based on the 2010 revised Jōyō kanji table for CJK ideographs, and the KS X 1001 encoding standard for Korean.
[*2] The Kana system in Japanese has diacritics, for a subset of the characters.
[*3] Languages which are written without spaces. Chinese and Japanese qualify as scriptio continua, while Korean is a special case where spacing rules are liberal and can be expressed without spaces in colloquial writing.

handing [*4] which artificially injects whitespace before and after CJK ideographs. This mechanism is not enabled for Korean.

3.3 OOV Mitigation

As OOV was a much more prevalent problem in the context of word-based methods, it has been investigated further than OOV in subword-based methods.

In the context of word-based models, pre-processing the input with stemming and lemmatization was a common practice, both to reduce OOV and the size of the vocabulary. Additionally, novel methods such as dictionary-based post-processing [17] and distributional representation based substitution [12] have been proposed.

However, in the context of subword tokenization this has not been actively investigated, aside from Ref. [27], which proposes adding new words to the vocabulary, and Ref. [20], which is our work.

For our experiments, we used four CJK 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. We intentionally chose sentiment analysis datasets, as the pre-trained model used (bert-base-multilingual-cased) was trained on Wikipedia and book corpora, and a domain shift to user-written content had a higher likelihood of suffering from OOV due to words that are unlikely to appear in well-formed content. Theoretically, Korean is expected to suffer the most, as the BERT tokenizer does not have special case handling, hence is susceptible to the greedy merging of the underlying WordPiece tokenizer.

4. Proposed Method

In this section, we propose a method to mitigate OOV without training a new model. This is based on a hypothesis that OOV has adversarial effects on task performance, which we also verify through experiments in Section 5.2. Our method is implemented as a modification of the BERT tokenizer. In all mitigation experiments, we compare with and without additional pre-training.

The BERT tokenizer is modified to support a secondary vocabulary which points new words to existing words for our experiments. This modified tokenizer is used instead of the original tokenizer in a 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 OOV subword surface cases 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. Whenever applicable, we use the previously computed frequency of the OOV tokens to prioritize frequent OOV tokens over rare cases. 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 additional pre-training and evaluate against the baseline.

Additional pre-training is the process of using the task corpus to train the model under a masked language modeling task, which is a form of additional pre-training, but against domain corpora. Here, the model is trained to fill in a masked portion of a given passage, given the context. Formally, this is called a cloze task and is the same process used to train BERT initially.

This additional pre-training intends to adapt the model so that it learns the changes in the vocabulary introduced by our mitigations, as the model has never seen the new subwords. This also helps the model better learn adequate representations that are better suited for the task domain. If the surrogate is assigned to a subword from a different language, for example, when using unseen subwords, this process is crucial. As this does not require an annotated corpus, it is also possible to make the model more robust by providing extra corpora.

Substitution to mitigate OOV has been studied in Ref. [12]. This method depends on part-of-speech tagging or a secondary corpus and model for similarity computation, challenging to apply in a subword model. Our approach's significance is that it works for subword models and its practical applicability, as only a downstream task corpus and a pre-trained model is required.

4.1 Surrogated Tokens

Surrogates, simply put, map 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 assigns multiple subwords to the same embedding, which is a trade-off that limits the utility of the proposed method for generation tasks. As surrogates are only assigned once, to perform generation tasks when a subword is polysemic, one would need to use an auxiliary binary classifier to determine which subword the prediction actually is. This is not required for tasks that do not require generation, such as classification.

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 $A \rightarrow X$

[*4] https://github.com/google-research/bert/blob/master/tokenization.py#L251

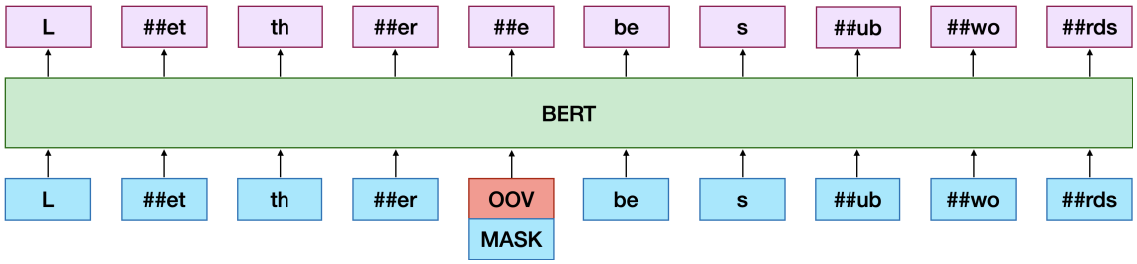


Fig. 2 Here, using the masked language modeling task from BERT, the OOV tokens are replaced with a mask to be predicted. The predictions are used as surrogates in an OOV-mitigated model.

6C10	汐	汦	汇	汙	汽	汕	汞	汗	汗	汙	污	汛	汜	汝	汞	江
AC10	감	갑	값	갓	갔	강	갖	갘	갘	같	갚	갛	개	객	갞	갰

Fig. 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.

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.

4.1.1 Character Distance

This method selects the surrogate with the shortest Unicode codepoint distance from the OOV subword, limited to subword tokens within the vocabulary of the same length. In this process, we perform an exhaustive search, formulated as the following.

$$\underset{w \in W'}{\operatorname{argmin}} |\operatorname{ord}(v) - \operatorname{ord}(w)|_1$$

In the formula above, v is the OOV subword, and W' is a subset of the vocabulary W which satisfies UTF-8 character level length equality $|v| = |w|$ for $w \in W$. ord is the Unicode ordinal conversion function.

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 as seen in **Fig. 3**. 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.

4.1.2 Unseen Subwords

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. In our experiments, this was implemented by overwriting the existing unseen subword to the target subword. This allows guaranteed reconstruction of the original text, making it usable for generation tasks, but at the cost of the embeddings being assigned to ones that the model has not seen in the context.

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.

4.1.3 Masked Language Model

The masked language model-based method uses BERT's masked language head to generate surrogate proposals, as illustrated in **Fig. 2**. 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 prediction later. This results in deterministic surrogate mappings.

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, OOV subwords that were not assigned a surrogate are assigned to the candidate with the lowest frequency. This method has the highest computation cost, as it requires inference on the model.

4.2 Additional Tokens

Here, we add new tokens to the vocabulary and increase the model size, motivated by prior work [29]. As this increases the network parameters, these are used as a secondary baseline to be compared with surrogates.

4.2.1 Random Initialization

After adding the missing subword to the vocabulary, then the corresponding embedding is randomly initialized. This is analogous to how a model is commonly initialized, and also how new tokens are added to an existing vocabulary.

4.2.2 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 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.

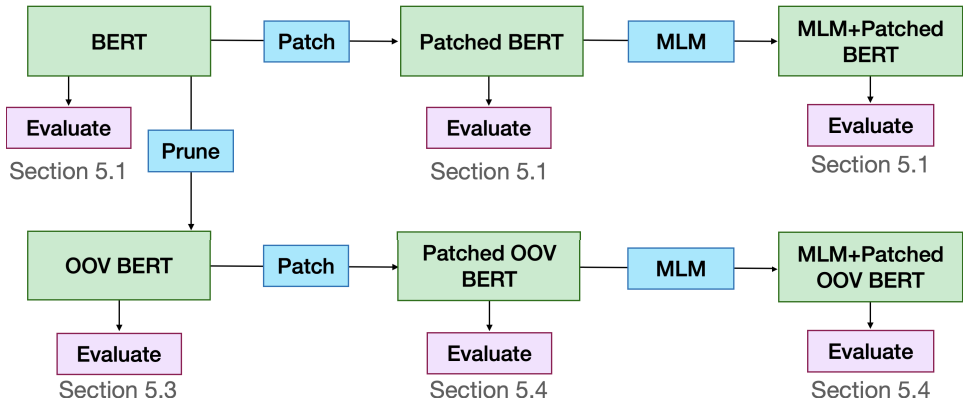


Fig. 4 The experiment pipeline of our work. Evaluations have been labeled with the corresponding section where the results are disclosed. MLM here is the process of performing additional pre-training through a masked language modeling task.

5. Datasets

5.1 Naver Sentiment Movie Corpus

The Naver Sentiment Movie Corpus [*5] (NSMC) [3] 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 on the pre-trained BERT model was 30.1% due to a large number of typos and the domain gap.

5.2 Japanese Twitter Sentiment Analysis

As a second validation target language, we used a subset [*6] of a Japanese Twitter dataset [25] [*7], which is a sentiment analysis task with five possible labels. The subset contains 20 K Tweets and 2 K Tweets, respectively, for training and test. We observed that a large portion of the OOV was from emojis during analysis, resulting in an OOV rate of 25.1% on the pre-trained BERT model.

5.3 Chinese News Sentiment Analysis

The INEWS dataset is part of the ChineseGLUE [*8] dataset. The input is a short sentence from a news article, and the label is one of three labels denoting the tone of the sentence. This is also a sentiment analysis task, with a split of 5 K train and 1 K validation, and an OOV rate of 20.1% on the pre-trained BERT model.

5.4 KorQuAD 1.0

KorQuAD 1.0 [*9] is a Korean version of the SQuAD [22] reading comprehension task. The task involves answering a question given a passage of text, and consists of 10 K passages with 66 K 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% on the pre-trained BERT model. For this task, additional pre-training was omitted to prevent the model from memorizing answers. We added this additional task to validate our method against a low-OOV task.

[*5] https://github.com/e9t/nsmc
[*6] https://github.com/cynthia/japanese-twitter
[*7] http://www.db.info.gifu-u.ac.jp/data/Data_5d832973308d57446583ed9f
[*8] https://github.com/chineseGLUE
[*9] https://korquad.github.io/

6. Experiments

To validate the effectiveness of our method proposed in the previous section, we perform multiple experiments against multiple CJK datasets in the upcoming sections. To thoroughly evaluate the effects of our proposed scheme, we validate against both real and synthetic setups, using the different mitigation schemes explained in Section 4. The high-level flow of all experiments we do here work is explained in **Fig. 4**. We compare the effects of different methods using a pre-trained multilingual BERT (bert-base-multilingual-cased). Each method was tested with fine-tuning, including a masked language modeling (additional pre-training) task, or by fine-tuning only against the task. Task-level fine-tuning was included in every experiment to ensure fairness and is done by attaching a task head and training the downstream task model. This allows the model to learn how to accomplish the task while adapting itself to produce better representations for the task. For our experiments, we limited additional pre-training to the task corpus to make the experiments reproducible with only the task datasets.

All experiments that involved training the model were trained for three epochs. The full list of hyperparameters used for the experiments is listed in Table A·1, in this paper's appendix. Every experiment in the upcoming section was run five times each, with the random seed fixed to an integer value of the run number in the range of [1..5]. The runs are then compared to the baseline scores to observe the statistical significance of the different scores for each method. For the significance test, we performed a dependent t-test for paired samples, following the guidelines in Ref. [9]. We used a p-value of $p < 0.05$ to determine statistical significance and a fixed seed (42) for any random algorithm to make the results deterministic, which guarantees reproducibility, as can be seen in **Table 2**. The evaluation was done with the reference implementation [*10] from Ref. [9].

6.1 Results on Task Datasets

The evaluation was done through the SST-2 GLUE task metrics [28] for the sentiment analysis tasks, and EM/F1 evaluation from the SQuAD metrics for KorQuAD, as the two tasks are com-

[*10] https://github.com/rtmdrr/testSignificanceNLP

Table 2 Scores across five runs, accompanied with statistical significance compared to baseline. Statistically significant points ($p < 0.05$) have been underlined in the p-values. Acc denotes accuracy and Std denotes standard deviation. Add models have more parameters. +MLM is with additional pre-training on the task corpus. Results for KorQuAD have been scaled down by 100, and are without additional pre-training.

		NSMC (ko)		Twitter (ja)		INEWS (zh)		KorQuAD (ko)	
Model	**Value**	**Acc@+MLM**	**Acc**	**Acc@+MLM**	**Acc**	**Acc@+MLM**	**Acc**	**EM**	**F1**
BERT (Baseline)	Mean	0.8824	0.8785	0.7284	0.7192	0.8138	0.8074	0.7037	0.9005
	Std	0.0017	0.0006	0.0041	0.0058	0.0064	0.0047	0.0016	0.0013
Add (Transfer)	Mean	0.8916	0.8844	**0.7319**	0.7223	0.8116	0.8082	0.7097	0.9030
	Std	0.0007	0.0006	0.0040	0.0060	0.0022	0.0082	0.0023	0.0011
	p-value	0.0002	0.0000	0.1091	0.1623	0.2599	0.4437	0.0091	0.0041
Add (Random)	Mean	**0.8928**	0.8848	0.7310	0.7211	**0.8186**	**0.8106**	**0.7098**	0.9029
	Std	0.0006	0.0004	0.0046	0.0041	0.0049	0.0065	0.0034	0.0018
	p-value	0.0000	0.0000	0.2263	0.1639	0.1280	0.0601	0.0128	0.0248
Char. Distance	Mean	0.8926	**0.8855**	0.7304	**0.7238**	0.8122	0.8092	0.7094	**0.9031**
	Std	0.0009	0.0013	0.0037	0.0024	0.0097	0.0070	0.0026	0.0019
	p-value	0.0001	0.0005	0.1499	0.0108	0.3152	0.1567	0.0115	0.0358
Unseen Subwords	Mean	0.8922	0.8846	0.7304	0.7225	0.8142	0.8102	0.7037	0.9013
	Std	0.0002	0.0013	0.0039	0.0038	0.0079	0.0065	0.0017	0.2112
	p-value	0.0000	0.0000	0.1554	0.0649	0.4403	0.1441	0.5000	0.2934
Masked LM	Mean	0.8915	0.8842	0.7307	0.7225	0.8100	0.8090	0.7089	0.9027
	Std	0.0009	0.0006	0.0043	0.0058	0.0063	0.0047	0.1647	0.1283
	p-value	0.0004	0.0002	0.1103	0.1219	0.1451	0.2801	0.0177	0.0614

patible. Each model used the same dataset and training parameters as the baseline, only with different OOV mitigation methods. The results of these experiments are in Table 2.

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

6.1.1 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. As seen in Table 2, we can indeed observe that regardless of the mitigation method, OOV mitigation, in general, improves accuracy and the improvements are statistically significant. 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 performance improves by domain adaptation through additional pre-training because the initial embeddings are not representative of the subword in context. As this dataset had the most significant gains in performance, we investigated the positive and negative examples in **Fig. 5**. As we have observed in **Table 3**, there were more cases which improved with our method. However, we also observed that negative cases emerge from additional pre-training, such as the samples in Fig. 5, some of which we suspect can be attributed to surrogates being assigned to a different language's Unicode page.

6.1.2 Japanese Twitter Sentiment Analysis

This corpus showed a high OOV rate due to the frequent occurrence of emoji in the text, and improper normalization of Unicode punctuation. We observe similar patterns with the results from NSMC. Generally, we see only minor improvements, except for character distance – which was statistically significant. We observed that character distance assigned surrogates to Korean characters [*11].

Table 3 Quantified improvements and regressions in performance across the different tasks with samples affected by OOV. The results are from the best scoring Character Distance models compared to best baseline models for each task.

Dataset	Regressed	Improved	Delta
NSMC	392	528	136
KorQuAD	64	79	15
Twitter	21	32	11
INEWS	11	11	0

6.1.3 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 Table 3, not only does the improved cases cancel out, looking at the OOV cases we considered the difference to be training noise. We hypothesize that small size of the dataset is likely to have contributed to the negative results.

6.1.4 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, although some methods produced p-values which were considered statistically significant. The small delta can most likely be attributed to the relatively low OOV rate and omission of additional pre-training.

Given that our experiments' results demonstrate that mitigat-

[*11] This would have been appropriate to demonstrate with examples, but due to the Twitter license agreement, reproducing the original text in this paper was not possible.

Type	Model	Text
Positive	Input	어릴때 **재밌게** 봤던 영화~
	Translation	A movie I **enjoyed** when I was young
	Baseline	어릴때 **[UNK]** 봤던 영화 ~
	Patched	어릴때 **재밋게** 봤던 영화 ~
	Input	**정맖재밋다**
	Translation	**Really interesting**
	Baseline	**[UNK]**
	Patched	**정말재밋다**
Positive (w/Bad Patches)	Input	**줬나 재미없다ㅅ1ㅂ** 영화보다가 **존건생ㅇ 처음림이랗그유치할수가**;; 평점 **ㅁㄷ고** 보다가 **개났ㄲ임**
	Translation	**Really boring. First time I fell asleep** during a movie. **Really lame. Tricked** by review scores.
	Baseline	**[UNK] [UNK]** 영화보다가 **[UNK] [UNK]** ; ; 평점 **[UNK]** 보다가 **[UNK]**
	Patched	**있나 재미없다고1ㅣ** 영화보다가 **존건생ヴ 처음림이랙그유치할수가** ; ; 평점 **ㅗン고** 보다가 **개났ㅌ임**
Negative	Input	유갓서비드가 **훨잼씀**
	Translation	"You got served" **is much more interesting**
	Baseline	유갓서비드가 **[UNK]**
	Patched	유갓서비드가 **훨쟁씀**
	Input	실질적으로 일반인들이 탈수있는차를 소개해달라니까 뭔 폴쉐911이야 **ㅋㅋㅋㅋㅋㅋ뭐하자는거임 ㅋㅋ**
	Translation	They introduce a Porsche 911 as a car for an average person? **(Laugh) What are they thinking?**
	Baseline	실질적으로 일반인들이 탈수있는차를 소개해달라니까 뭔 폴쉐911이야 **[UNK] [UNK]**
	Patched	실질적으로 일반인들이 탈수있는차를 소개해달라니까 뭔 폴쉐911이야 **・・・・・・뭐하자는거임 ・・**

Fig. 5 Positive, positive (with bad patches) and negative examples with the proposed method applied. Negative cases are surrogate assignments which had adversarial effects on performance, and positive is the opposite. Positive with bad patches is a special case where the assignment looks incorrect, but contributed positively to performance. The OOV surfaces have been marked in bold.

ing OOV improves task performance, in the next section, we explore if our method can recover performance in high-OOV models, which we have synthetically created through initial OOV and performance correlation experiments.

6.2 Effects of OOV on Task Performance

In the previous section, we demonstrated the effects of our method on different tasks and languages. These experiments were conducted based on the hypothesis that OOV has an adversarial impact on task performance. In this section, we artificially induce OOV on a pre-trained model through vocabulary pruning and correlate the OOV rate to task performance. With these synthetic OOV models, we use one of the tasks to investigate how OOV affects task performance in a BERT model. Following this, we apply our scheme to these synthetic models to verify if our proposed method is effective at recovering the performance of a broken model.

In this section, we investigate the correlation between OOV and task performance by evaluating task performance using the baseline BERT (bert-base-multilingual-cased) model, then compare the results of that to models with varying OOV rates.

We use the three methods to eliminate the most frequent words, the least frequent words, and random sampling. We compare different methods to ensure fairness, as the different methods exhibit different scenarios of how an OOV can be introduced in a downstream task. NSMC was chosen because it was the largest dataset we had for our experiments, and we assumed that the larger the task corpus is, the more likely it will have a diverse vocabulary, hence being more susceptible to OOV.

For the frequency computation, we used two datasets. The first dataset we used is the kosentences [*12] corpus. This corpus is a Korean corpus cleansed of Wiki markup from multiple publicly available Wiki dumps. As we only use the Wikipedia part of kosentences, we will refer to the corpus as KoWiki in this paper. We considered this to be a good approximation of what the backbone model (bert-based-multilingual-cased) was initially trained with. This is because almost every large-scale pre-train corpus contains Wikipedia in some form. For this case, the frequency table was initialized with every Korean subword in the model, and the frequencies against the KoWiki corpus were updated on the frequency table. Subwords in the model's vocabulary, but not in the KoWiki corpus, were kept at a 0. The second dataset used was the actual task corpus, as using the task corpus is the most effective way to introduce OOV artificially.

This experiment intends to correlate the relation between OOV rate and task performance to confirm our initial hypothesis. It is worth noting that as we do not train a model from scratch, this is an approximation and not an accurate representation of what a pre-trained model's vocabulary would have due to the properties of subword tokenization depending on the character level n-gram distribution. This trade-off was made for computational efficiency reasons, as pre-training, a new model requires a significant amount of computing power, and for our experiments, we will need to train 42 models, which was computationally infeasible.

In our experiments, we prune subwords from the frequency table in different ratios – for our experiments, we chose 0.1%, 1%, 5%, 10%, 20%, and 50% as the target ratios. 20% and 50% are

[*12] https://github.com/cynthia/kosentences

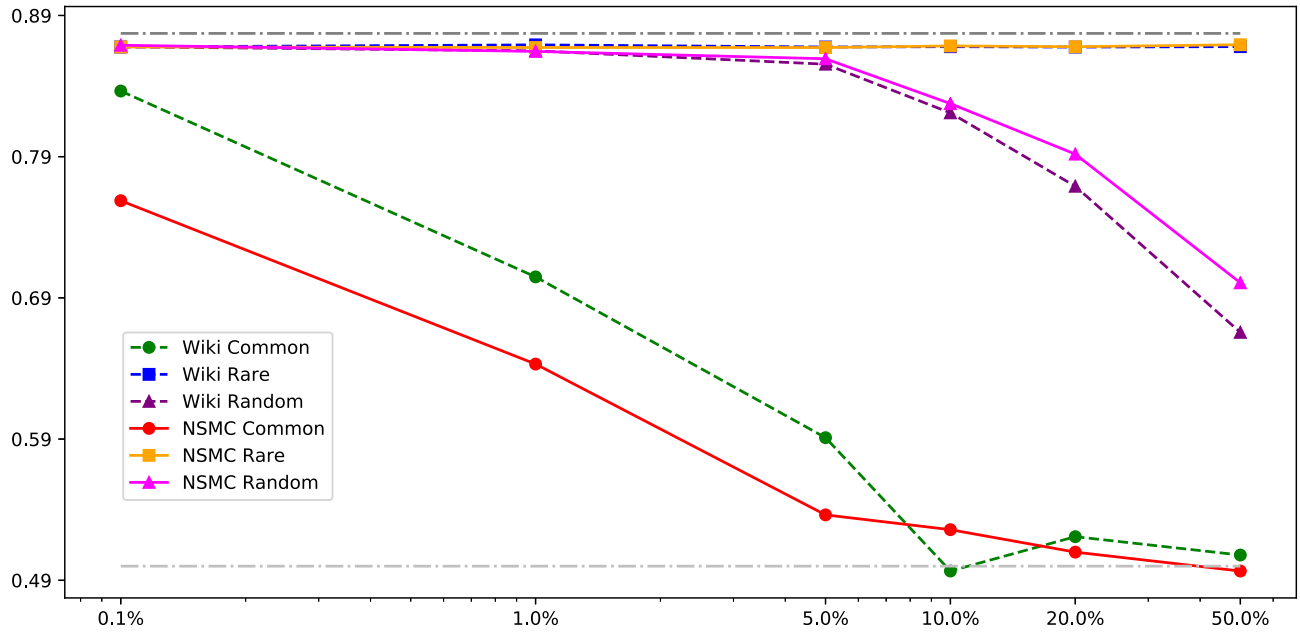


Fig. 6 Effects of task performance caused by artificially induced OOV. All three pruning strategies have been compared in this plot.

used to test extreme scenarios, to a point where it is likely that the model predictions can be considered equivalent to random choice. We use three different strategies for pruning the vocabulary, which we discuss in the subsections below. The ratio here is the ratio of words of the frequency table's vocabulary we prune from the vocabulary and should not be mistaken with the OOV ratio discussed in the datasets section.

6.2.1 Common Words

Removing the most frequent words is not a common scenario in any form, especially when it comes to a pre-trained model setup. Ranking the vocabulary in order of frequency, we prune the vocabulary from the top ranking (most frequent) word based on the ratio to be pruned. For example, in a 1,000 word vocabulary with a 5% prune rate, the end result will be a model that is missing 50 of the most frequent words. This was chosen to demonstrate the extreme cases of unusable models, for instance, if a language that was expected to be supported was accidentally omitted from the training data.

6.2.2 Rare Words

This method was chosen to simulate a scenario where the corpus was sampled, or character coverage was reduced due to computational constraints. As least frequent subwords in a corpus will be omitted from the vocabulary, we consider this a rough approximation of what would happen when trade-offs are made due to the computational limitations. The process is the same as common words, but in this case, pruning is in order of least frequent words.

6.2.3 Random Words

In random words, we randomly eliminate subwords from the vocabulary. The subwords list in the frequency table is used to select target subwords to remove from the vocabulary. Based on the target subword list, we randomly choose a word for removal and evaluate the performance.

This is also another approximation of the consequences of computational feasibility trade-offs, as with least frequent words. As the distribution of subword frequency is expected to follow Zipf's law, even with random removal, we assume that the probability of an infrequent subword being chosen for deletion is inversely proportional to the frequency of the given subword.

6.3 Correlating Task Performance with OOV

The results of these experiments are summarized in **Fig. 6**, accompanied by the full results in Table A·2. An important point to note here is that as this is a balanced, binary classification task, it is unlikely that a model's accuracy can go significantly below 0.5. As it converges towards 0.5, we can consider the model's output to be equivalent to an equidistributed binary random number generator, hence a random model.

Based on the experiment results, the first straightforward observation we made is that removing rare words does not affect task performance at all, regardless of how many are removed. Analyzing the removed words, we observed that the removed words were mostly words from a different language, which we suspect will not have substantial contributions to task performance. On the other hand, the other two methods used for pruning affect accuracy, especially as the ratio increases.

We observed that pruning common words had immediate effects, especially using the KoWiki corpus – as the effects are apparent even at 0.1%. This is because the vocabulary of the frequency table of KoWiki is larger than that of NSMC, 0.1% pruned more subwords than the NSMC frequency table, which had a smaller vocabulary. Removal of common words can have devastating effects, as, without a matching suffix form of a subword, the tokenizer's greedy will fail. In both cases, we can see that starting from around 5%; the model converges towards a random model's performance. In these worst-case scenarios, we observed that the model's input had more OOV than actual subwords. In many cases, input to the model exclusively consisted

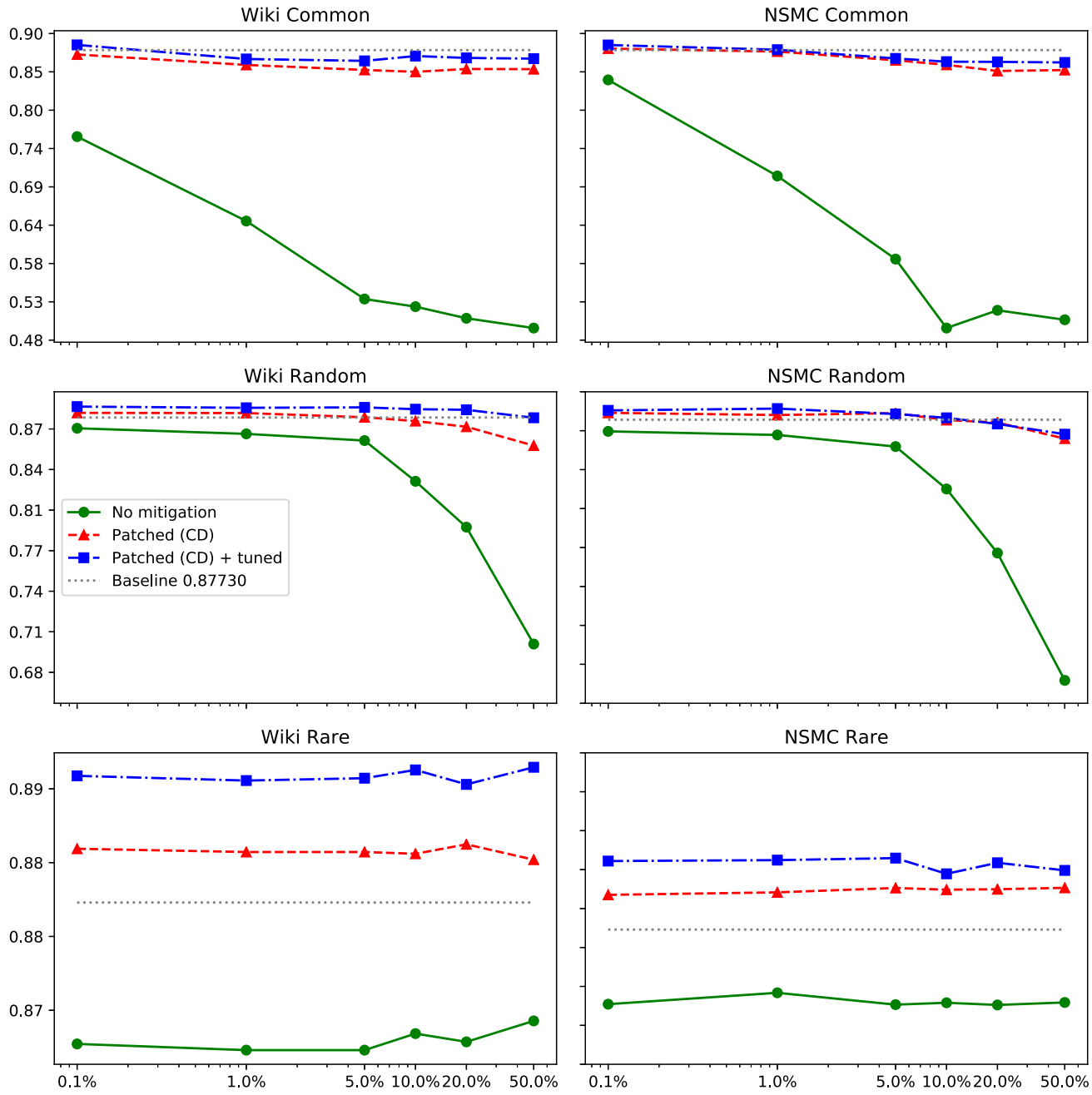


Fig. 7 Performance recovery under different OOV rates. Note that the plot range for the y-axis is different between common, random, and rare.

of OOV tokens.

Random pruning, on the other hand, tended to have a slower effect on task performance. This is expected, as the probability of pruning a common subword is lower than the probability of pruning a less common word. We can still observe noticeable performance decreases on both KoWiki and NSMC starting from 5%. Unlike common, the model did not end up in a state comparable to random choice.

Even when scaled up to 50%, pruning rare subwords did not have significant effects on the performance. This was somewhat unexpected, as we initially hypothesized it to affect the task performance with that many words removed. The reason turned out to be that even at 50%, most of the rare words only appeared once, and only a small portion (less than 5%) of these rare words were Korean – which explains the minimal effect on performance.

We use the models from this experiment with the same protocol we proposed to mitigate OOV for the recovery experiments. Among the multiple methods proposed, we use character distance (CD), as it was shown to be effective while being computationally efficient, which allowed us to experiment with many different configurations.

6.4 Recovery with Proposed Method

The results of this experiment are visualized in **Fig. 7**, and the full results are in Table A·3. The trends we observed in the original mitigation NSMC experiments repeat here. A model that has been both through additional pre-training and OOV-patching consistently outperformed a model without additional

Table 4 OOV Rate tested on the task datasets used for our experiments with language-specific models. Rates have been rounded the first decimal digit. (0.0% is any value under 0.05%.)

Dataset	Model	OOV Tokens	Total Tokens	Token Rate	OOV Sentences	Total Sentences	Sentence Rate
NSMC	bert-base-multi	81,603	5,185,891	1.5%	60,151	200,000	30.1%
NSMC	KR-BERT	360	4,773,732	0.0%	336	200,000	0.1%
KorQuAD	bert-base-multi	14,159	5,134,799	2.8%	8,569	144 K	5.9%
KorQuAD	KR-BERT	5,978	4,396,060	1.4%	2,393	144 K	1.7%
Twitter	bert-base-multi	10,310	985,345	1.0%	5,518	22,000	25.1%
Twitter	cl-tohoku-base-v2	26,566	951,286	2.8%	10,165	22,000	46.2%
INEWS	bert-base-multi	2,570	158,212	1.6%	1,278	6,355	20.1%
INEWS	bert-base-chinese	2,338	158,065	1.5%	1,119	6,355	17.6%

pre-training. In this particular setup, we hypothesize that additional pre-training contributions in models with higher OOV rates can be attributed to the fact that many subwords have now been mapped to semantically different words, so the model has to learn the structure of the text nearly from scratch. However, our results suggest that the proposed method can even be effective at improving performance in high-OOV conditions, such as a model that was pre-trained on corpora extremely disparate from the target task's domain.

While the model does not fully manage to recover to the best-case performance fully, we also observed that extreme case models such as those with comparable performance to a random model could also recover quite well. However, in these extreme case models, we observed that due to the high amount of surrogates needed, the model started borrowing subwords from other languages, from the CJK ideograph block as a surrogate for Korean subwords.

We do not have any theoretical proof of why extreme cases like this can also recover near-baseline performance. We hypothesize that without additional pre-training, the model's representations lack semantic or contextual information; hence it acts as a random embeddings model. This model can still be used to classify data with a classification head. With additional pre-training, the model re-learns the structure from the input, only with different embeddings, and due to the surrogate assignment being exclusive in our method, the model can adapt to inherent structure from the task corpus.

7. Applicability to Other Models

7.1 Multilingual Models

While our experiments are limited to BERT, the method can be applied to any model. Generally, our proposed method is most effective when applied to greedy merging tokenizers such as WordPiece, which is used by both BERT and ELECTRA [4]. This is due to the fact that greedy merging results in whole chunks of text being lost during tokenization, as we have observed in the Fig. 5 examples.

However, our method is applicable to most subword tokenization methods, such as Byte-pair Encoding (BPE), used by the multilingual model XLM [15], and SentencePiece [14], used by another multilingual model, XLM-R [5] also can benefit from this. The effects will be less significant since both tokenizers are not greedy. The expected effect is a diversification of the UNK token by re-assigning it to different subwords instead of all OOV tokens being mapped to a single embedding. This is expected to make it easier to train. In an actual byte-level [*13] subword tokenization, such as used in GPT-2 [21], our method is not expected to have any gains as there will always be a byte-level fallback.

7.2 Monolingual Models

Following the discussion on our method's applicability to different multilingual models, we also investigated whether or not OOV is also a phenomenon in language-specific models. As our method depends on the occurrence of OOV in the first place, if there is a low OOV rate, the contributions of OOV mitigation are also expected to be minor. To investigate this, we used three separate monolingual models for each language.

For Chinese, we used the official BERT Chinese model (bert-base-chinese) released as part of the pre-trained models in Ref. [8], with the BERT tokenizer, and for Japanese we used [*14]. Finally, for Korean, we used KR-BERT [16] with Normalization Form Compatibility Decomposition (NFKD) [*15] pre-processing. The subcharacter decomposition is similar to the work proposed in Ref. [19] and makes this method much more robust against OOV. Each of the monolingual models was used to tokenize the respective language dataset compared with the multilingual model used in this work. The results are disclosed in **Table 4**.

We observed that Korean, which is the most effective language to our scheme, we can see that the amount of OOV tokens in this model is extremely low. Due to this, it is unlikely to have adversarial effects on performance. While the OOV token ratio was still above 1% for KorQuAD, most of this turned out to be caused by subwords in a foreign language (e.g., CJK Ideographs), which is unlikely to have severe effects as it is assumed that the reader does not necessarily have to comprehend this from the passage to produce an answer [*16] for the task.

Japanese, on the other hand, showed an increase in OOV. This is likely because the pre-training corpus was Wikipedia, which is well-formed text lacking colloquial writing, and Emojis, common in data sourced from social networks. In Chinese, there was very little difference as with the multilingual model, so the effects of applying our method are likely to be the same as a multilingual model.

[*13] Byte-pair Encoding is commonly misrepresented, as while the original method does operate at byte-level, current applications all operate at Unicode character level.
[*14] https://github.com/cl-tohoku/bert-japanese
[*15] This model does not work if this normalization is omitted.
[*16] We confirmed that none of the answers expected an answer in a different language from the dataset.

8. Conclusions

In this work, we investigate the correlation between OOV and task performance in the context of transfer learning. With differing OOV rates, we confirm our hypothesis that OOV directly affects the performance of a model in a transfer setup, to the point that it makes the model comparable to a model that is randomly choosing answers.

After demonstrating examples and the effects of OOV triggered information loss with evaluation performance in a task under a transfer learning setup, we propose multiple mitigating OOV methods 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 at the model level, especially when paired with additional pretraining. Additionally, with the same experiments, we also confirm that tasks with lower OOV suffer less than tasks with higher OOV.

We further explore the applicability of our work in extreme cases and use the high-OOV models we used to test our hypothesis, combined with the proposed mitigation can recover the model's capabilities and conclude that in a transfer learning setup, tokenization serves a significant role in a pre-trained model's capabilities.

8.1 Future Work

Additionally, one of our work's limitations is that most of the surrogate methods cannot be used for generative tasks, as tokens are replaced with other tokens. We expect future work to explore potential solutions for this limitation. Finally, while we provided a hypothesis, the reason why BERT can still perform using unseen subwords for classification is not an answered question and warrants further investigation.

Acknowledgments This paper is based on results obtained from a project, JPNP18002, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). Part of the compute used for the experiments were provided by Odd Concepts Inc.

The authors also thank Won Ik Cho, Tatsuya Hiraoka, Sakae Mizuki, Sho Takase, and Angela Smiley for their suggestions and insightful discussions.

References

[1] Aji, A.F., Bogoychev, N., Heafield, K. and Sennrich, R.: In Neural Machine Translation, What Does Transfer Learning Transfer?, *Proc. 58th Annual Meeting of the Association for Computational Linguistics*, pp.7701–7710, Association for Computational Linguistics (online), DOI: 10.18653/v1/2020.acl-main.688 (2020).

[2] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D.: Language models are few-shot learners (2020).

[3] Cho, W.I., Moon, S. and Song, Y.: Open Korean Corpora: A Practical Report, *Proc. 2nd Workshop for NLP Open Source Software* (*NLP-OSS*), pp.85–93, Association for Computational Linguistics (online), DOI: 10.18653/v1/2020.nlposs-1.12 (2020).

[4] Clark, K., Luong, M.-T., Le, Q.V. and Manning, C.D.: ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, *ICLR* (2020) (online), available from ⟨https://openreview.net/pdf?id=r1xMH1BtvB⟩.

[5] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L. and Stoyanov, V.: Unsupervised Cross-lingual Representation Learning at Scale, *Proc. 58th Annual Meeting of the Association for Computational Linguistics*, pp.8440–8451, Association for Computational Linguistics (online), DOI: 10.18653/v1/2020.acl-main.747 (2020).

[6] Conneau, A. and Lample, G.: Cross-lingual Language Model Pretraining, *Advances in Neural Information Processing Systems 32*, Wallach, H., Larochelle, H., Beygelzimer, A., d′Alché-Buc, F., Fox, E. and Garnett, R. (Eds.), pp.7057–7067, Curran Associates, Inc. (2019) (online), available from ⟨http://papers.nips.cc/paper/8928-cross-lingual-language-model-pretraining.pdf⟩.

[7] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1* (*Long and Short Papers*), pp.4171–4186, Association for Computational Linguistics (online), DOI: 10.18653/v1/N19-1423 (2019).

[8] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1* (*Long and Short Papers*), pp.4171–4186, Association for Computational Linguistics (online), DOI: 10.18653/v1/N19-1423 (2019).

[9] Dror, R., Baumer, G., Shlomov, S. and Reichart, R.: The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing, *Proc. 56th Annual Meeting of the Association for Computational Linguistics* (*Volume 1: Long Papers*), pp.1383–1392, Association for Computational Linguistics (online), DOI: 10.18653/v1/P18-1128 (2018).

[10] Gillick, D., Brunk, C., Vinyals, O. and Subramanya, A.: Multilingual Language Processing From Bytes, *Proc. 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp.1296–1306, Association for Computational Linguistics (online), DOI: 10.18653/v1/N16-1155 (2016).

[11] Howard, J. and Ruder, S.: Universal Language Model Fine-tuning for Text Classification, *Proc. 56th Annual Meeting of the Association for Computational Linguistics* (*Volume 1: Long Papers*), pp.328–339, Association for Computational Linguistics (online), DOI: 10.18653/v1/P18-1031 (2018).

[12] Kolachina, P., Riedl, M. and Biemann, C.: Replacing OOV Words For Dependency Parsing With Distributional Semantics, *Proc. 21st Nordic Conference on Computational Linguistics*, pp.11–19, Association for Computational Linguistics (2017) (online), available from ⟨https://www.aclweb.org/anthology/W17-0202⟩.

[13] Kudo, T. and Richardson, J.: SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, *EMNLP 2018 – Proc. Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (online), DOI: 10.18653/v1/d18-2012 (2018).

[14] Kudo, T. and Richardson, J.: SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing, *Proc. 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp.66–71, Association for Computational Linguistics (online), DOI: 10.18653/v1/D18-2012 (2018).

[15] Lample, G. and Conneau, A.: Cross-lingual Language Model Pretraining, *Advances in Neural Information Processing Systems* (*NeurIPS*) (2019).

[16] Lee, S., Jang, H., Baik, Y., Park, S. and Shin, H.: KR-BERT: A Small-Scale Korean-Specific Language Model (2020).

[17] Luong, T., Sutskever, I., Le, Q., Vinyals, O. and Zaremba, W.: Addressing the Rare Word Problem in Neural Machine Translation, *Proc. 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (*Volume 1: Long Papers*), pp.11–19, Association for Computational Linguistics (online), DOI: 10.3115/v1/P15-1002 (2015).

[18] Merity, S., Keskar, N.S. and Socher, R.: Regularizing and Optimizing LSTM Language Models, arXiv preprint arXiv:1708.02182 (2017).

[19] Moon, S. and Okazaki, N.: Jamo Pair Encoding: Subcharacter Representation-based Extreme Korean Vocabulary Compression for Efficient Subword Tokenization, *Proc. 12th Language Resources and Evaluation Conference*, pp.3490–3497, European Language Resources Association (2020) (online), available from

⟨https://www.aclweb.org/anthology/2020.lrec-1.429⟩.
[20] Moon, S. and Okazaki, N.: PatchBERT: Just-in-Time, Out-of-Vocabulary Patching, *Proc. 2020 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pp.7846–7852, Association for Computational Linguistics (online), DOI: 10.18653/v1/2020.emnlp-main.631 (2020).
[21] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I.: Language Models are Unsupervised Multitask Learners, Technical Report (2018).
[22] Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P.: SQuad: 100,000+ questions for machine comprehension of text, *EMNLP 2016 – Proc. Conference on Empirical Methods in Natural Language Processing* (online), DOI: 10.18653/v1/d16-1264 (2016).
[23] Sennrich, R., Haddow, B. and Birch, A.: Neural machine translation of rare words with subword units, *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 – Long Papers* (online), DOI: 10.18653/v1/p16-1162 (2016).
[24] Stratos, K.: A Sub-Character Architecture for Korean Language Processing, *Proc. 2017 Conference on Empirical Methods in Natural Language Processing*, pp.721–726, Association for Computational Linguistics (online), DOI: 10.18653/v1/D17-1075 (2017).
[25] Suzuki, Y.: Filtering Method for Twitter Streaming Data Using Human-in-the-Loop Machine Learning, *Journal of Information Processing*, Vol.27, pp.404–410 (online), DOI: 10.2197/ipsjjip.27.404 (2019).
[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I.: Attention is All you Need, *Advances in Neural Information Processing Systems*, Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. and Garnett, R. (Eds.), Vol.30, pp.5998–6008, Curran Associates, Inc. (2017).
[27] Wan, Z., Wan, X. and Wang, W.: Improving Grammatical Error Correction with Data Augmentation by Editing Latent Representation, *Proc. 28th International Conference on Computational Linguistics*, pp.2202–2212, International Committee on Computational Linguistics (2020) (online), available from ⟨https://www.aclweb.org/anthology/2020.coling-main.200⟩.
[28] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S.: GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding, *Proc. 2018 EMNLP Workshop Blackbox NLP: Analyzing and Interpreting Neural Networks for NLP*, pp.353–355, Association for Computational Linguistics (online), DOI: 10.18653/v1/W18-5446 (2018).
[29] Wang, H., Yu, D., Sun, K., Chen, J. and Yu, D.: Improving Pre-Trained Multilingual Model with Vocabulary Expansion, *Proc. 23rd Conference on Computational Natural Language Learning* (*CoNLL*), pp.316–327, Association for Computational Linguistics (online), DOI: 10.18653/v1/K19-1030 (2019).
[30] Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M. and Dean, J.: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, *CoRR*, Vol.abs/1609.08144 (2016) (online), available from ⟨http://arxiv.org/abs/1609.08144⟩.

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. For the OOV correlation and recovery tasks, we optimized the sequence length and batch size parameter specifically to the task to maximize VRAM usage for faster experimentation. The exact hyperparameters are disclosed in **Table A·1** [*17].

The masking probability for the MLM task was set to 0.15 for additional pre-training. We did not use whole word masking for our experiments.

A.2 Environment and Computation Cost

The experiments in this paper were run on two different environments. The additional pre-training and accompanied evaluation experiments were executed on a shared rt_G.small instance on the ABCI compute cluster [*18]. An rt_G.small node has six segregated CPU cores from a Xeon Gold 6148, a Tesla V100 GPU with 16 GB VRAM, and 60 GBs of memory. The training data and experimental code was streamed from a shared GPFS mount. Each experiment requires a 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.

The OOV correlation and recovery NSMC experiments were executed on a desktop computer with a Ryzen 9 3900XT 12-core processor, RTX3090 GPU with 24 GBs of VRAM, and 64 GBs of memory. The training data and experimental code were on a local Phison E16 NVMe drive. Both tasks required the same amount of compute budget, and the average runtime for each experiment was approximately 40 minutes with the hyperparameter optimizations used above.

A.3 Experiment Result Tables

The results obtained from all of the experiments, including those omitted from the plots, are in **Tables A·2** and **A·3** respectively.

[*17] The parameters used will use 23.5 GBs out of 24 GB of available VRAM when training using IEEE754 half-precision floating point tensors.

[*18] https://abci.ai/

Table A·1 Hyperparameters used to train each of the downstream task models.

Task	Optimizer	Adam ϵ	LR	GradAccum	Weight Decay	Length	Batch Size	Epochs
Additional Pre-training	Adam	1e-8	5e-5	1	0.0	512	6	3
OOV Correlation (NSMC)	Adam	1e-8	2e-5	1	0.0	160	160	3
Mitigation (GLUE)	Adam	1e-8	2e-5	1	0.0	512	10	3
Question Answering (KorQuAD)	Adam	1e-8	3e-5	1	0.0	512	12	3
OOV Recovery (NSMC)	Adam	1e-8	2e-5	1	0.0	160	160	3

Table A·2 Experiment results demonstrating the effects of OOV in an artificial setup. The percentages are the ratio of words removed based on the frequency table computed with different data. Method is the different methods used for pruning.

Frequency Table	Method	0%	0.1%	1%	5%	10%	20%	50%
KoWiki	Common	0.87730	0.75882	0.64312	0.53632	0.52584	0.50994	0.49654
KoWiki	Rare	0.87730	0.86772	0.86730	0.86730	0.86842	0.86786	0.86928
KoWiki	Random	0.87730	0.86892	0.86456	0.85934	0.82758	0.79182	0.70068
NSMC	Common	0.87730	0.83650	0.70486	0.59100	0.49656	0.52088	0.50788
NSMC	Rare	0.87730	0.86772	0.86918	0.86766	0.86790	0.86762	0.86794
NSMC	Random	0.87730	0.86784	0.86494	0.85550	0.82104	0.76904	0.66556

Table A·3 Recovery experiment results. Patched is with mitigation, Patched+MLM is with mitigation and additional pre-training.

Frequency Table	Sampler	Mitigation	0%	0.1%	1%	5%	10%	20%	50%
KoWiki	Common	None	0.87730	0.75882	0.64312	0.53632	0.52584	0.50994	0.49654
		Patched (CD)	0.88390	0.87132	0.85702	0.85014	0.84756	0.85146	0.85120
		Patched (CD) + MLM	0.88850	0.88446	0.86514	0.86256	0.86918	0.86662	0.86562
KoWiki	Rare	None	0.87730	0.86772	0.86730	0.86730	0.86842	0.86786	0.86928
		Patched	0.88390	0.88094	0.88072	0.88072	0.88060	0.88124	0.88020
		Patched (CD) + MLM	0.88850	0.88588	0.88556	0.88572	0.88628	0.88530	0.88646
KoWiki	Random	None	0.87730	0.86892	0.86456	0.85934	0.82758	0.79182	0.70068
		Patched	0.88390	0.88092	0.88078	0.87752	0.87452	0.87012	0.85542
		Patched (CD) + MLM	0.88850	0.88582	0.88490	0.88522	0.88380	0.88334	0.87710
NSMC	Common	None	0.87730	0.83650	0.70486	0.59100	0.49656	0.52088	0.50788
		Patched	0.86830	0.87942	0.87530	0.86340	0.85716	0.84872	0.85000
		Patched (CD) + MLM	0.88850	0.88432	0.87788	0.86590	0.86146	0.86122	0.86040
NSMC	Rare	None	0.87730	0.86772	0.86918	0.86766	0.86790	0.86762	0.86794
		Patched	0.86830	0.88176	0.88208	0.88264	0.88242	0.88248	0.88266
		Patched (CD) + MLM	0.88850	0.88610	0.88622	0.88648	0.88446	0.88588	0.88490
NSMC	Random	None	0.87730	0.86784	0.86494	0.85550	0.82104	0.76904	0.66556
		Patched	0.86830	0.88288	0.88112	0.88264	0.87702	0.87506	0.86194
		Patched (CD) + MLM	0.88850	0.88488	0.88634	0.88196	0.87882	0.87388	0.86554

Sangwhan Moon received his master's degree in computer science from the Georgia Institute of Technology in 2017. He is a director of engineering at Odd Concepts Inc. and an elected member of the World Wide Web Consortium (W3C) Technical Architecture Group. He also is a standardization expert for the Korean Telecommunications Technology Association. His research interests include natural language processing, computer vision, machine learning, and information retrieval.

Naoaki Okazaki is a professor in School of Computing, Tokyo Institute of Technology, Japan. Prior to this faculty position, he worked as a post-doctoral researcher in University of Tokyo (2007–2011), andas an associate professor in Tohoku University (2011–2017). He is also a senior scientific research specialist of Ministry of Education, Culture, Sports, Science and Technology (MEXT) and a visiting research scholar of the Artificial Intelligence Research Center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST). His research areas include Natural Language Processing (NLP), Artificial Intelligence (AI), and Machine Learning.

(Editor in Charge: *Yuki Arase*)