

Subword Tokenization of Noisy Housing Defect Complaints for Named Entity Recognition

Kahyun Jeon¹ and Ghang Lee²

¹Department of Architecture and Architectural Engineering, Yonsei University, Republic of Korea
jeonkh0310@yonsei.ac.kr, glee@yonsei.ac.kr

Abstract –

In domain-specific named entity recognition (NER), the out-of-vocabulary (OOV) problem arises due to linguistic features and rare vocabulary. OOV problem is particularly challenging in agglutinative languages such as Korean. The irregular decomposition of morphemes makes it difficult to represent all of them in language model dictionaries, resulting in poor NER performance. Subword tokenization which segments a word into atomic tokens that are no longer divided can be one of the possible solutions. In the construction industry, existing NER methods do not effective on housing defect complaints which contain many rare words, including jargon, slang, and typos. To address this challenge, we propose subword tokenization algorithms that can mitigate OOV problems based on considering linguistic features and pre-trained language models (PLMs). The primary objective of this study is to identify the optimal NER performance by comparing different subword tokenization methods depending on the language models used. For domain-specific NER, we defined and used 23 defect-specific named entity tags for dataset labelling. We then experimented with a total of three state-of-the-art language models: one SentencePiece-based and two WordPiece-based subword tokenization models. The results demonstrate that the SentencePiece-based Korean Bidirectional Encoder Representations from Transformers (KoBERT) outperformed the two WordPiece-based language models (multilingual-BERT and Korean Efficiently Learning an Encoder that Classifies Token Replacements Accurately (KoELECTRA)) with an F1 score of 84.7%. The proposed method is expected to improve not only NER but also other downstream tasks that involve using Korean documents in the construction industry.

Keywords –

Out of vocabulary (OOV), Subword tokenization; WordPiece; SentencePiece; Named entity recognition (NER); Construction defect management

1 Introduction

In natural language processing (NLP), many studies have shown that the out-of-vocabulary (OOV) problem degrades the performance of downstream tasks, such as named entity recognition (NER), speech recognition, and neural machine translation [1–4]. OOV occurs because domain-specific terms, non-standard words, or typos in a test dataset do not exist in the vocabulary of a training corpus [5].

A representative example of unstructured noisy text data is housing defect complaints, which include numerous non-standard words and words that rarely appear in construction text. When defect complaints are automatically analyzed, such non-standard and rarely used words are often divided into unintended ways, and consequently, the original meaning is lost. Moreover, these incorrectly tokenized characters are recognized as ‘unknown’ tokens, tagged as ‘[UNK]’ in the NER task results. The ‘[UNK]’ token refers to the failure to decode corresponding tokens based on the embedded vocabulary sets in each pre-trained language model (PLM). Therefore, the OOV problems become more serious as non-standard words used only in specific domains, as well as typos and syntax errors, increase.

Among many previous efforts to reduce OOV, subword tokenization is one of the promising solutions [3,6–8]. Subword tokenization refers to the segmentation of words into smaller tokens that can be aggregated or decomposed based on the principles of subword tokenization algorithms [9]. According to these principles, frequently appearing terms should not be segmented, but rarely used combinations of characters should be decomposed [10].

The most widely used subword tokenization methods in transformer-based PLMs are the WordPiece and SentencePiece algorithms [11,12]. While WordPiece tokenizes input text by finding the sequence of characters that can make the longest word, SentencePiece tokenizes input text by predicting the most common token that appears after the previous token [12]. Thus, WordPiece is sensitive to the wordlist of a given language while

SentencePiece is language-independent [13].

In this study, we investigated the effect of the difference between WordPiece and SentencePiece algorithms on NER performance. Three pre-trained language models—Multilingual Bidirectional Encoder Representations from Transformers (mBERT), Korean Efficiently Learning an Encoder that Classifies Token Replacements Accurately (KoELECTRA), and Korean BERT (KoBERT)—were fine-tuned using a defect complaint corpus through transfer learning. Each pre-trained model has its own embedded tokenizer with two different subword tokenization algorithms, either WordPiece or SentencePiece. To perform NER based on housing defect complaints, we also defined defect-specific named entity tags in the pre-processing step. The NER performance was evaluated using the F1 score and accuracy.

The paper is organized as follows. In Section 2, related works on solutions to OOV problems and subword tokenization methods are reviewed. In Section 3, the research methods are described in detail. In Section 4, the results are presented with some discussion, and in Section 5, study limitations and suggestions for future work are provided.

2 Related Work

2.1 OOV Problems and Solutions

Fundamentally, OOV problems occur because a language model is dependent on a training dataset [14]. During the language model training process, if the words do not appear in the training dataset, the language model cannot learn the embedding vectors. Consequently, OOV problems degrade the system performance. More severe OOV problems occur in low-resource languages (e.g., non-alphabetical languages) or web-generated text, including slang, coinage, and emojis [4]. To overcome these problems, several solutions have been proposed.

The easiest and simplest method is to eliminate OOV as noise [14]. However, this method is rarely used, owing to the risk of losing important words. Another solution is to use spell-check algorithms, such as the Peter Norvig algorithm [15] and the Levenstein distance [16]. However, spell-check algorithms cannot solve semantic OOV problems caused by jargon or slang.

Previous studies have proposed deep learning-based typo embedding methods [17] or language-independent architecture of robust word vectors [18]. However, these methods were not able to separate irregular morphemes of agglutinative words into meaningful units. Consequently, they could not be applied to solve OOV problems in agglutinative languages, such as Korean.

Some studies focused on dealing with OOV problems considering the linguistic features of Korean [14,19–21].

One of the specific linguistic features of Korean that previous studies mainly focused on is a sub-character, called ‘jamo’, which consists of 14 consonants and 10 vowels. However, although those studies have proposed high performance on tokenization regarding sub-characters, the proposed methods are limited to use for the tokenizer the existing transformer-based language models. A comparison of the previous efforts to mitigate OOV problems is shown in Table 1.

Table 1. Comparison of the existing methods that mitigate OOV problems

Previous methods	Advantage	Disadvantage	Ref.
Remove OOV	Easy	Losing information	[14]
Spell-check	Easy, Accurate	Not effect for semantic errors	[15,16]
Deep-learning-based	Robust on typos	Not effect for rich-morpheme languages	[17,18]
Language-specific OOV handling	Specific to language features	Limited to transformer-based language models	[14,19–21]

To overcome the previous methods’ limitations, and solve the OOV problem while utilizing most state-of-the-art (SOTA) language models based on the transformer architecture, subword tokenization—a method for segmenting words into smaller units—has emerged [4]. The smaller units can be Unicode or characters, which can no longer be divided based on the specific algorithm [22]. Various subword tokenization algorithms that have been used by integrating transformer-based language models are described in the following section.

2.2 Subword Tokenization Algorithms

There are several types of subword tokenization methods, such as byte pair encoding (BPE) [3], unigram language model (ULM) [6], WordPiece [4], and SentencePiece [13].

BPE is a basic subword tokenization algorithm, which finds pairs of characters that appear the most sequentially and combines them into one word [3]. BPE was originally introduced to compress data [23], after which the algorithm was proposed as an OOV solution for neural machine translation [3]. In the iterative process of BPE, a set containing a pair or token and a token’s frequency is generated. Then, each token is divided into character levels by a pre-tokenizer according to the space. In each iteration, the most frequent character pair is

merged until the desired or pre-defined vocabulary size is reached. The core idea of BPE is to consider text as a sequence of bytes rather than a sequence of characters. Therefore, BPE can be applied to any type of character [3].

The WordPiece model is an expanded BPE algorithm. An ideal subword tokenizer divides low-frequency or morphologically complex words into smaller subword units, otherwise keeping high-frequency words as they are. The WordPiece tokenizer is used in BERT, DistillBERT, and ELECTRA [4]. The WordPiece model merges pairs of characters that increase the likelihood of a corpus, which is different from BPE, which uses word frequency. In other words, WordPiece algorithms choose to divide or preserve words depending on the direction that increases the probability of the language model the most.

A unigram language model (ULM) was proposed based on a subword tokenization algorithm that outputs multiple subword tokenization with probabilities [6]. The ULM supposes that each subword occurs independently, and consequently. Therefore, each subword occurrence probability can be estimated by maximizing the expected likelihood. A ULM is usually used with SentencePiece in transformer-based language models to increase system performance.

SentencePiece is an unsupervised subword tokenizer and detokenizer [13,24], and was developed based on ULM [6] and BPE [3] subword regularization algorithms. Unlike all other algorithms, which require pre-tokenization based on whitespace, SentencePiece directly processes the input text without pre-tokenization. Instead of pre-tokenization, SentencePiece can be trained using raw sentences that include whitespace. In addition, the original whitespace can be preserved by replacing it with an underscore ('_'; represented by U+2581 in Unicode) in the subword tokenization stage. Conversely, the subword tokens can be detokenized without any ambiguity caused by whitespace. For example, the complaint 'wall paper torn' can be segmented as ['_w', 'all', '_p', 'ap', 'er', '_t', 'or', 'n'] by the SentencePiece tokenizer. This shows that SentencePiece considers a sentence as a Unicode sequence rather than characters. This principle makes SentencePiece work on non-space or space-free languages. This is the most significant difference from other subword tokenization algorithms because it makes the SentencePiece language independent. As SentencePiece supports on-the-fly processing through Python and Tensorflow Library API, it can be easily integrated and customized with other frameworks. SentencePiece has been widely adopted in many SOTA language models, such as ALBERT [25], XLNet [26], and T5 [27]. A comparison of subword tokenization methods is shown in Table 2.

After conducting a thorough review of previous

efforts, we have identified several research gaps. First, there exist multiple effective subword tokenization algorithms that are dependent on the linguistic features of a given language. The language-dependent tokenizer requires separate training for every language it may not perform as well in cross-lingual tasks. Second, the choice of subword tokenization algorithm is determined during the pre-training process of the internal tokenizer of the PLMs. Third, the various subword tokenization approaches employed by different PLMs can have a significant impact on the performance of downstream tasks.

Table 2. Comparison of subword tokenization methods

Cate gory	BPE [3]	ULM [6]	Word Piece [4]	Sentence Piece [13]
SA	Char, Unicode	Char	BPE	BPE, ULM
FP	Word frequency	Likeli- hood	Likelihood	BPE- dropout
VS	Enlargeme nt	Pre- defined	Enlarge- ment	Pre- defined
PT	Require	Require	Require	No
OS	Yes	Yes	Google internal	Yes
PL	No	No	N/A	Yes
HT	Yes	Yes	Yes	Yes

* SA: Supported algorithms; FP: Functional principle; VS: Vocab-Size; PT: Pre-tokenization required; OS: Open source; PL: Python library available; HT: HuggingFace tokenizer available.

In light of these research gaps, it is crucial to consider the linguistic features, such as the agglutinative nature of the Korean language, when performing NER using transformer-based PLMs. Fourth, NER tasks with domain-specific text data and their respective subword tokenization algorithm have been rarely studied. Therefore, this study aims to identify the most effective PLM and subword tokenization algorithm for domain-specific NER tasks for construction defect management.

3 Research Methods

The research flow is depicted in Figure 1. For this study, over 90,000 defect complaints were collected from several collective housing construction sites in South Korea. After performing deduplication and removing abnormal data, a dataset of 69,750 complaints was used for training and validation. Additionally, we created a separate ground truth test dataset consisting of 4,566 complaints gathered from different construction sites. The test dataset was constructed through cross-validation

by three researchers, who possess extensive defect inspection experience at construction sites.

After pre-processing and tokenizing, the collected data were labeled with defect named entity tags for NER. Labeling was performed on a rule basis using a dictionary of defective object names that corresponded to 23 tags defined in advance. The first labeling results were cross-validated by three researchers who perform defect inspections at construction sites.

Subsequently, the labeled dataset was divided into a training dataset for transfer learning and a testing dataset for the NER performance evaluation at a ratio of 9:1. Before transfer learning, the PLM uses each tokenizer to perform subword tokenization of the input text. In the test phase of the fine-tuned language model, the same subword tokenization was applied. The NER results are compared with the F1 score and accuracy as evaluation indices.

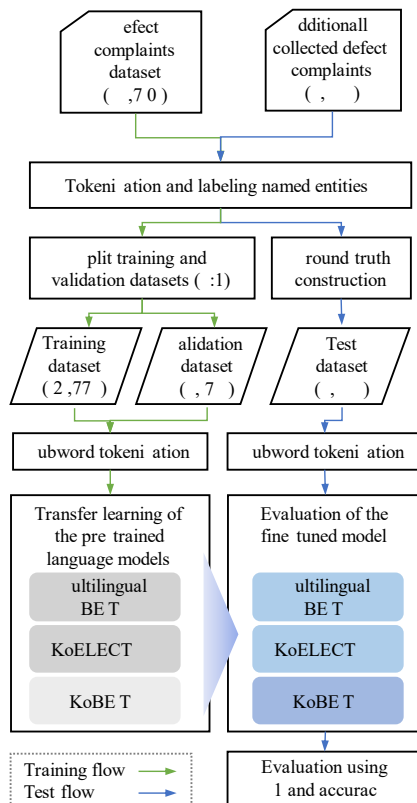


Figure 1. Research flow

3.1 Tokenization and Labeling Dataset

As Korean is an agglutinative language, most Korean words consist of ‘a stem of the word (a meaning part)’ and ‘a particle or an ending (a grammatical part).’ For example, ‘물이 새는(water leak)’ can be decomposed to the following combination: ‘물(water: stem)’+

‘ㅇ](particle)’ + ‘새(-다)(leak: stem)’+ ‘(ending)’. The morpheme analysis-based tokenization method is required to separate and process the meaning parts and the grammatical function parts.

To achieve this, we used Korean-specific NLP libraries, including MeCab-Ko [28], a Korean version of the original MeCab [29], for morpheme analysis and tokenization. In the first pre-processing step, the date or time expressions, ‘07/07/2021’ and ‘07:37’, were processed and extracted using regular expression rules, and other punctuation and symbols were removed. Then, the rest of the pure text part was tokenized using the MeCab-Ko tokenizer. The pure text part of the defect complaints was tokenized, and the part of speech (POS) was tagged. We extracted not only nouns but also verbs, adjectives, and adverbs because they also represent defect phenomena: for example, ‘skew(-ed)’ or ‘tilt(-ed)’.

After tokenization, all tokens should be labeled with an NE tag, information that we want to extract. Moreover, it is necessary to label the entire dataset to construct the training dataset and the ground truth. Before labeling the NE tags, we need to define defect-related NE categories. Unlike existing NER methods that have general NE tags, such as the name of the country, city, or person, domain-specific NE tags should be defined to recognize the specialized terms in a certain domain. In this study, 17 tags were defined based on Omniclass Table 22 [30]. The categories include structure work (STR), waterproofing (WPF), electrical work (ELC), door and window (DNW), cabinet work (FUR), stonework (STM), paper hanging work (PAP), flooring (FLR), painting and finishing (FNS), tiling (TLE), masonry (MSN), home appliance work (APL), mechanical systems (MEC), gas fitting work (GAS), fire protection work (FIR), miscellaneous (MIS), and elevator work (ELV). In addition, five meta-data categories were defined to identify the date and time (TME), the name of the room or space (SPC), a specific part of the space, such as ‘above’ or ‘left side’ (LOC), the type of request (REQ), the name of person or organization in charge of the repair work (WHO) and expressions that describe defect phenomena (DFT).

To label the dataset using 23 NE tags, we checked the tokenization error cases, which means a series of words that should not be decomposed are tokenized. For example, a multi-token entity consists of more than one word, such as ‘poor horizontal and vertical alignment’, which is tokenized as five words: ‘poor/ horizontal/ and/ vertical/ alignment’. If this multi-token entity is decomposed, the original meaning will be lost, and it cannot be properly recognized as a defect NE. As a solution, we performed tokenization and NE labeling according to the ‘inside-outside-beginning2 (IOB2) scheme’. The IOB2 scheme is a popular tagging scheme for recognizing a multi-token NE in an NER task [31]. ‘I’ indicates that the token is inside the NE, and ‘B’ indicates

the start token of the NE. ‘O’ indicates that the token belongs to none of the NE. ‘I’ and ‘B’ are joined in front of each NE tag (i.e., B-DFT, B-SPC, and I-SPC), and ‘O’ is used alone. The dataset was labeled based on the rule-based method, and then the automatically labeled dataset was cross-validated by three researchers who have defect inspection experience.

3.2 Pre-trained Language Models with Different Subword Tokenization Algorithms

The PLM has a substantial number of context vectors and vocabularies. Thus, it is easy to shift and use efficiently for specific tasks by fine-tuning based on the transfer learning rather than training the whole neural architecture from scratch. For specific domains (e.g., medical or construction), fine-tuning a PLM with a domain-specific corpus is recommended to achieve higher performance [32]. Therefore, we chose three PLMs and implemented them for NER transfer learning.

Three PLMs with different subword tokenizers were used to transfer learning for NER. The mBERT developed by Google and KoELECTRA have a WordPiece tokenizer, while KoBERT uses a SentencePiece tokenizer. As the dataset is Korean, we first selected mBERT [33] as the baseline model. We also selected KoBERT, a specifically trained BERT with the Korean dataset KoWiki composed of 54 million words [34]. KoELECTRA [35] was chosen to compare the performance of different language model algorithms between BERT and ELECTRA, which is known for exceeding BERT’s performance in terms of training speed with less data. The specifications of the three pre-trained language models are described in Table 3.

Table 3. Details of the pre-trained language models

Category	KoBERT	mBERT	Ko-ELECTRA
Tokenizer Algorithm	SentencePiece	WordPiece	WordPiece
Pre-trained Language	Korean	104 languages	Korean
Parameter	92M	110M	110M
Layers	12	12	12
Reference	[34]	[36,37]	[38,39]

3.3 Experiment Design

Subword tokenization was performed as a previous step for fine-tuning PLMs. The input text data are tokenized by the pre-trained tokenizer designated by each language model. In this process, the label of each token

was also expanded to the same number of subword tokens. Then, subword tokens and labels were converted to a numerical format using a pad sequence, and the attention mask was generated at the same time. We observed the subword tokenization results in the middle of the experiment process to determine how subword tokenization affects the final NER performance. To compare this with the case without subword tokenization, a deep neural network model, a bi-directional long-short-term-memory (bi-LSTM) network, was performed as a baseline.

All input texts were fit into the same fixed length of the 0.985 quantile value (top 1.5%) of the entire training dataset, because all the text lengths of the input data are different. In this process, the 0.985 quantile value was defined as the maximum length, one of the hyperparameters that determines the input vector shape. The same value of the 0.985 quantile of the input text length was applied to the input of the testing dataset in the evaluation.

In the experiments, all three PLMs were fine-tuned to only three epochs with a batch size of 32, an ‘adam’ optimizer with a learning rate of 2.E-05, and ‘parseCategoricalCrossentropy’ as for a loss function. It is important to note that batch size refers to the number of data points used to update the model’s parameters during a single forward/backward pass. The ‘adam’ optimizer, which stands for ‘adaptive moment Estimation’ is commonly used in deep neural networks for its training efficiency. Despite the default learning rate value of 1.E-03, we opted for a smaller value of 2.E-05 for enhancing generalization efficiency.

3.4 Evaluation Metrics

Accuracy and the F1 score were employed to evaluate the NER performance considering each performance of the NE tags. The total number of NE tags is 46, which is twice the unique 23 tags owing to the addition of ‘I’ and ‘B’ of the IOB2 scheme. Similar to the multi-class classification with imbalanced data distribution, the F1 score, a harmonic mean of precision and recall, is widely used for evaluating NER performance [40]. Accuracy refers to the ratio of the number of correctly predicted samples over the number of all samples regardless of class. Therefore, we measured both F1 score and accuracy to consider the imbalanced distribution of each NE tag. The evaluation metrics for integrating NER tags can be automatically obtained with a Python library for sequence labeling evaluation named ‘seqeval’ [41].

4 Results and Discussion

The NER performance results depending on the three PLMs with different subword tokenization algorithms are shown in Table 4.

Table 4. NER performance results for the deep neural network-based model and three PLMs with different subword tokenization methods

NER Models	SW-T*	F1 (%)	Acc. (%)
(Baseline) bi-LSTM	-	23.0	49.0
mBERT	WP	72.0	86.6
KoELECTRA	WP	72.4	86.3
KoBERT	SP	84.7	89.3

*SW-T: Subword tokenization algorithm; WP: WordPiece; SP: SentencePiece

For the NER performance, the KoBERT model using the SentencePiece algorithm as a subword tokenizer had the best F1 score (84.7%) and an accuracy of 89.3%. There was little difference in performance between the BERT and ELECTRA models using the WordPiece algorithm, but a significant difference in the F1 score, which was 11.7% lower than that of the KoBERT model. For the baseline, the bi-LSTM showed a very low performance even with the same training and test datasets except for the subword tokenization step.

Table 5. Subword tokenization results (translated into English from the original Korean words)

Word Tag	KoBERT	mBERT	KoELECTRA
Zendai	_Zendai	Zen###da##i	Zen##dai
	B-STN	○	B-STN
Osai	_Osai	O##sa##i	Osa##i
	B-DNW	○	B-DNW
Ventilating Fan	_Ventilating	[UNK]	Ventilating
	_Fan		##Fan
	B-SYS/ I-SYS	○	B-SYS/ I-SYS
Floating	_Float_ing	Flo##[UNK]	Floating
	B-DFT/ I-DFT	○	B-DFT
Horizontal Defect	_Horizontal	Horizon##tal#	Horizontal
	_defect	#de##fect	##defect
	B-DFT/ I-DFT	B-DFT	B-DFT/ I-DFT

To compare the subword tokenization results of three language models, we selected five words—zendai, osai, ventilating fan, floating, and horizontal defect—that frequently lead to tokenization errors when a morpheme-based tokenizer is used.

The results for subword tokenization with the corresponding NER tag relying on three PLMs are shown in Table 5. KoBERT with SentencePiece correctly

tokenized all words, but mBERT and KoELECTRA showed different results although they used the same WordPiece algorithm. In mBERT, the defect-related tokens were incorrectly tagged with the ‘O’ tag because the segmented tokens were recognized as [UNK].

The subword sequences tokenized by mBERT showed a smaller segmentation pattern similar to KoELECTRA. This result can be interpreted as KoELECTRA was trained one more time in Korean, and naturally, its vocabulary is larger; thus, the probability of preserving a word is greater than the probability of splitting a word.

5 Conclusion

The OOV problem deteriorates the overall NER performance because invalid tokens recognized as ‘unknown’ fail in named entity tagging. And this problem occurs more often in agglutinative language such as Korean due to its complex morphological feature. To mitigate this problem, subword tokenization methods have been widely used, which either decompose the word into subword units or aggregate in reverse based on the maximization of probability or likelihood between the sequence of the subword units. Based on these principles, most state-of-the-art transformer-based language models adopt WordPiece or SentencePiece, the representative subword tokenization algorithms, as their tokenizer.

In this study, we investigated differences in NER performance when we suggested that the different subword tokenization methods affect the overall downstream task. To validate this hypothesis, mBERT and KoELECTRA using WordPiece and KoBERT using SentencePiece were selected. In NER, each subword tokenization was applied before the fine-tuning and testing steps. This means that each language model obtains a different shape of the sequence of the tokens as an input, depending on the subword tokenization method.

As a result, this study confirmed that the KoBERT model using the SentencePiece tokenizer showed the best performance based on the F1 score (84.7%), as well as the most accurate tokenization results. In addition, SentencePiece showed more robust tokenization results in Korean than WordPiece. Although this study has limitations in not controlling for other factors that can affect performance, such as parameter optimization during transfer learning or skewed distribution of NE tags, they will be further investigated and adjusted to deliver more robust results in future studies. Another future study will include further validation of whether the direct training of SentencePiece using a domain-specific corpus improves downstream tasks.

Acknowledgements

This work was supported by a National Research Foundation of Korea (NRF) grant (No. 2021R1A2C3008209) and an Institute for Information and Communications Technology Promotion (IITP) grant (No. 2019-0-01559-001), both funded by the Ministry of Science and ICT (MSIT) of Korea.

References

- [1] N. Garneau, J.-S. Leboeuf, L. Lamontagne, Predicting and interpreting embeddings for out of vocabulary words in downstream tasks, (2019). <https://doi.org/10.48550/arXiv.1903.00724>.
- [2] Y. Goldberg, Neural Network Methods for Natural Language Processing, *Synthesis Lectures on Human Language Technologies*. 10 (2017) 1–309. https://doi.org/10.2200/S00762ED1V01Y201703_HLT037.
- [3] R. Sennrich, B. Haddow, A. Birch, Neural Machine Translation of Rare Words with Subword Units, (2016). <https://doi.org/10.48550/arXiv.1508.07909>.
- [4] M. Schuster, K. Nakajima, Japanese and Korean voice search, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012: pp. 5149–5152. <https://doi.org/10.1109/ICASSP.2012.6289079>.
- [5] Y. Pinter, R. Guthrie, J. Eisenstein, Mimicking Word Embeddings using Subword RNNs, (2017). <https://doi.org/10.48550/arXiv.1707.06961>.
- [6] T. Kudo, Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates, (2018). <https://doi.org/10.48550/arXiv.1804.10959>.
- [7] S. Moon, N. Okazaki, PatchBERT: Just-in-Time, Out-of-Vocabulary Patching, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020: pp. 7846–7852. <https://doi.org/10.18653/v1/2020.emnlp-main.631>.
- [8] S. Moon, N. Okazaki, Jamo Pair Encoding: Subcharacter Representation-based Extreme Korean Vocabulary Compression for Efficient Subword Tokenization, in: Proceedings of the 12th Language Resources and Evaluation Conference, European Language Resources Association, Marseille, France, 2020: pp. 3490–3497. <https://www.aclweb.org/anthology/2020.lrec-1.429> (accessed March 23, 2021).
- [9] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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), Association for Computational Linguistics, Minneapolis, Minnesota, 2019: pp. 4171–4186. <https://doi.org/10.18653/v1/N19-1423>.
- [10] Summary of the tokenizers, (n.d.). https://huggingface.co/docs/transformers/main/tokenizer_summary (accessed January 4, 2023).
- [11] A. Nayak, H. Timmapathini, K. Ponnalagu, V. Gopalan Venkoparao, Domain adaptation challenges of BERT in tokenization and sub-word representations of Out-of-Vocabulary words, in: Proceedings of the First Workshop on Insights from Negative Results in NLP, Association for Computational Linguistics, Online, 2020: pp. 1–5. <https://doi.org/10.18653/v1/2020.insights-1.1>.
- [12] R. Ma, Y. Tan, X. Zhou, X. Chen, D. Liang, S. Wang, W. Wu, T. Gui, Q. Zhang, Searching for Optimal Subword Tokenization in Cross-domain NER, (2022). <https://doi.org/10.48550/arXiv.2206.03352>.
- [13] T. Kudo, J. Richardson, SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing, *ArXiv:1808.06226 [Cs]*. (2018). <http://arxiv.org/abs/1808.06226> (accessed August 20, 2021).
- [14] O. Kwon, D. Kim, S.-R. Lee, J. Choi, S. Lee, Handling Out-Of-Vocabulary Problem in Hangeul Word Embeddings, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Association for Computational Linguistics, Online, 2021: pp. 3213–3221. <https://doi.org/10.18653/v1/2021.eacl-main.280>.
- [15] P. Norvig, How to Write a Spelling Corrector, (2007). <http://norvig.com/spell-correct.html> (accessed December 2, 2021).
- [16] K. Kukich, Techniques for automatically correcting words in text, *Acm Computing Surveys (CSUR)*. 24 (1992) 377–439.
- [17] B. Edizel, A. Piktus, P. Bojanowski, R. Ferreira, E. Grave, F. Silvestri, Misspelling Oblivious Word Embeddings, (2019). <https://doi.org/10.48550/arXiv.1905.09755>.
- [18] V. Malykh, V. Logacheva, T. Khakhulin, Robust Word Vectors: Context-Informed Embeddings for Noisy Texts, in: Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-Generated Text, Association for Computational Linguistics, Brussels, Belgium, 2018: pp. 54–63. <https://doi.org/10.18653/v1/W18-6108>.

- [19] S. Lee, H. Shin, The Korean Morphologically Tight-Fitting Tokenizer for Noisy User-Generated Texts, in: Proceedings of the Seventh Workshop on Noisy User-Generated Text (W-NUT 2021), Association for Computational Linguistics, Online, 2021: pp. 410–416. <https://doi.org/10.18653/v1/2021.wnut-1.45>.
- [20] S. Eo, C. Park, H. Moon, H. Lim, Research on Subword Tokenization of Korean Neural Machine Translation and Proposal for Tokenization Method to Separate Jongsung from Syllables, Journal of the Korea Convergence Society. 12 (2021) 1–7. <https://doi.org/10.15207/JKCS.2021.12.3.001>.
- [21] K. Park, J. Lee, S. Jang, D. Jung, An Empirical Study of Tokenization Strategies for Various Korean NLP Tasks, ArXiv:2010.02534 [Cs]. (2020). <http://arxiv.org/abs/2010.02534> (accessed January 11, 2021).
- [22] T. Mikolov, I. Sutskever, A. Deoras, H.-S. Le, S. Kombrink, J. Cernocky, Subword language modeling with neural networks, Preprint (<http://www.fit.vutbr.cz/Imikolov/Rnnlm/Char>. Pdf). 8 (2012). <http://www.fit.vutbr.cz/Imikolov/rnnlm/char.pdf>.
- [23] P. Gage, A new algorithm for data compression, C Users Journal. 12 (1994) 23–38.
- [24] SentencePiece, (2023). <https://github.com/google/sentencepiece> (accessed January 3, 2023).
- [25] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, ArXiv:1909.11942 [Cs]. (2020). <http://arxiv.org/abs/1909.11942> (accessed May 21, 2021).
- [26] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, Q.V. Le, XLNet: Generalized Autoregressive Pretraining for Language Understanding, ArXiv:1906.08237 [Cs]. (2020). <http://arxiv.org/abs/1906.08237> (accessed May 21, 2021).
- [27] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, (2020). <https://doi.org/10.48550/arXiv.1910.10683>.
- [28] Y. Lee, Y. Yoo, Eunjeon / mecab-ko — Bitbucket, (2013). <https://bitbucket.org/eunjeon/mecab-ko-dic/src/master/> (accessed August 9, 2021).
- [29] T. Kudo, Mecab: Yet another part-of-speech and morphological analyzer, SourceForge. (2006). <https://sourceforge.net/projects/mecab/> (accessed August 9, 2021).
- [30] C. OmniClass, OmniClass Table 22- Work Results approved, (2013). www.omniclass.org.
- [31] H.-C. Cho, N. Okazaki, M. Miwa, J. Tsujii, Named entity recognition with multiple segment representations, Information Processing & Management. 49 (2013) 954–965. <https://doi.org/10.1016/j.ipm.2013.03.002>.
- [32] J. Li, A. Sun, J. Han, C. Li, A Survey on Deep Learning for Named Entity Recognition, ArXiv:1812.09449 [Cs]. (2020). <http://arxiv.org/abs/1812.09449> (accessed February 2, 2021).
- [33] J. Devlin, BERT-Base: Multilingual Cased, GitHub Repository. (2019). <https://github.com/google-research/bert/blob/master/multilingual.md>.
- [34] H. Jeon, SKT-KoBERT: Korean BERT pre-trained cased, GitHub Repository. (2020). <https://github.com/SKTBrain/KoBERT>.
- [35] J. Park, KoELECTRA: Pretrained ELECTRA Model for Korean, GitHub Repository. (2020). <https://github.com/monologg/KoELECTRA>.
- [36] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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), Association for Computational Linguistics, Minneapolis, Minnesota, 2019: pp. 4171–4186. <https://doi.org/10.18653/v1/N19-1423>.
- [37] J. Devlin, BERT-Base: Multilingual Cased, GitHub Repository. (2019). <https://github.com/google-research/bert/blob/master/multilingual.md>.
- [38] K. Clark, M.-T. Luong, Q.V. Le, C.D. Manning, ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, ArXiv:2003.10555 [Cs]. (2020). <http://arxiv.org/abs/2003.10555> (accessed August 11, 2021).
- [39] J. Park, KoELECTRA: Pretrained ELECTRA Model for Korean, GitHub Repository. (2020). <https://github.com/monologg/KoELECTRA>.
- [40] P. Hühthwohl, R. Lu, I. Brilakis, Multi-classifier for reinforced concrete bridge defects, Automation in Construction. 105 (2019) 102824. <https://doi.org/10.1016/j.autcon.2019.04.019>.
- [41] H. Nakayama, seqeval: A Python framework for sequence labeling evaluation, (2018). <https://github.com/chakki-works/seqeval>.