# Comparison of Korean Preprocessing Performance according to Tokenizer in NMT Transformer Model

Geumcheol Kim and Sang-Hong Lee
Department of Computer Science & Engineering, Anyang University, Anyang-si, Republic of Korea
Email: rmacjf9988@naver.com, shleedosa@gmail.com

***Abstract*—Mechanical translation using neural networks in natural language processing is making rapid progress. With the development of natural language processing model and tokenizer, accurate translation is becoming possible. In this paper, we will create a transformer model that shows high performance recently and compare the performance of English Korean according to tokenizer. We made a traditional neural network-based Neural Machine Translation (NMT) model using a transformer and compared the Korean translation results according to the tokenizer. The Byte Pair Encoding (BPE)-based Tokenizer showed a small vocabulary size and a fast learning speed, but due to the nature of Korean, the translation result was not good. The morphological analysis-based Tokenizer showed that the parallel corpus data is large and the vocabulary is large, the performance is higher regardless of the characteristics of the language.**

***Index Terms*—translation, tokenizer, neural machine translation, natural language processing, deep learning**

## I. Introduction

As Natural Language Processing (NLP) field that is applied with deep learning recently recorded high performance, research in related field is actively going on [1]-[3]. In particular, as the Transformer model using self-attention only introduced in "Attention is all you need," the traditional neural network-based Neural Machine Translation (NMT) solved long-term-tem-dependency of the previous RNN and LSTM learning speed and direction, interest in NMT using the Transformer model has increased [4]-[6]. BERT, GPT, and XLNet, the latest NLP models using Transformer models, have various NMT models that present the best State of the Art (SOTA) in the system training, evaluation and analysis algorithm Generative Language Understanding Evaluation (GLUE) metrics for natural language comprehension [7]-[9].

This performance enhancement has a significant impact on performance, not only on neural network-based models, but also the process of preprocessing parallel coppers data. In particular, the tokenize process, in which sentences are divided into token (words or letters), varies greatly depending on the characteristics of each language. Recently, we are using tokenizer based on morphological analysis that identifies grammatical structures such as root, prefix, and verb, and Tokenizer based On Byte Pair Encoding (BPE), a data compression technique that can reduce problems (Out of Vocabulary, OOV) that do not exist in learning.

Because the performance of these tokenizer varies from language to language, it seems necessary to find tokenizer that fits the Korean language. The composition of this paper aims to introduce the Transformer model and create the English-Korean NMT model to enhance the performance of Korean translation by comparing the performance according to tokenizer.

## II. Experimental and Preprocessing

### A. Tokenizer

Tokenize is called tokenize to cut sentences into appropriate to tokenize. Tokenize is after all a sentence divided into semantic units. This is to create vocabulary during this preprocessing and map that to a computer-readable, real-time vector.

Tokenizer based on morphological analysis is to analyze the form of a sentence and separate the sentences by form factor. Morphology is divided into the smallest units with a certain meaning and the part-of-speech (pos) of the sentence. Examples include various morphological analyzers such as NLTK, Khaii, Hannanum, Kkma, Komoran, Mechab and Twitter.

TABLE I. Tokenizer Based on Morphological Analysis

| Hannanum | Kkma | Komoran | Mecab | Twitter |
|---|---|---|---|---|
| 나/N | 나/NP | 나/NP | 나/NP | 나/Noun |
| 는/J | 는/JX | 는/JX | 는/JX | 는/Josa |
| 밥/N | 밥/NNG | 밥/NNG | 밥/NNG | 밥/Noun |
| 을/J | 을/JKO | 을/JKO | 을/JKO | 을/Josa |
| 먹/P | 먹/VV | 먹/VV | 먹/VV | 먹는/Verb |
| 는다/E | 는/EPT | 는다/EC | 는다/EC | 다/Eomi |
| | 다/EFN | | | |

Manuscript received May 6, 2020; revised September 29, 2020.


doi: 10.12720/jait.11.4.228-232

As shown in Table I, the sentence '나는 밥을 먹는다' is divided into '나/NP + 는/JX + 밥/NNG + 을/JKO + 먹/VV + 는다/EC' [10]. However, because tokenizer based on morphological analysis predicts that analysis results are based on dictionaries that have been established, incorrect analysis results can be produced if errors or newly coined words are included.

The BPE-based Tokenizer mitigated OOV problems by creating a dictionary for Subword Unit based on BPE algorithm, which is a data compression technique [11]. BPE is the way we create bottom up set of words in letter units and eventually get results by merging unigrams with the highest frequency.

Table II shows a set of words after 10 iterations of BPE algorithm. In the first stage, dictionary's 'l o w -> lo o o w, l o w e r -> ro o o o we er r ...' is divided into pairs of backs and find the frequency of each pair. And create a set of words with the most frequency. Repeat this n times to complete the set of words. Even if the new word lowest comes in with a complete set of words, lowest can be created in combination with 'low' and 'est' to prevent OOV.

TABLE II. TOKENIZER BASED ON BPE [3]

| | n times | vocabulary |
|---|---|---|
| Dictionary<br>Word (frequency)<br>l o w : 5<br>l o w e r : 2<br>n e w e s t : 6<br>w I d e s t : 3 | 1 | l, o, w, e, r, n, w, s, t, i, d, es |
| | 2 | l, o, w, e, r, n, w, s, t, i, d, es, est |
| | 3 | l, o, w, e, r, n, w, s, t, I, d, es, est, lo |
| | … | |
| | 10 | l, o, w, e, r, n, w, s, t, I, d, es, est, lo, low, ne, new, newest, wi, wid, widest |

## B. Transformer

Previously, sequence data were mostly processed in the recurrent model [12]. The recurrent model uses the $t^{th}$ input and $(t-1)^{th}$ hidden state to produce the $t^{th}$ output. In this way, when the sequential nature of the sentence is maintained, but prior long-term information needs to be considered for the $t^{th}$ result, there is a long-term-dependency problem where the gap between sentences increases, the earlier information cannot be considered.

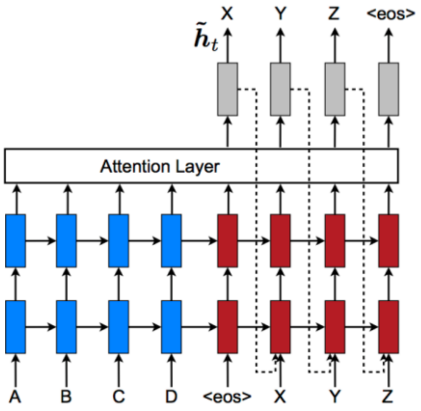



Figure 1. Feeding hidden state as input to decoder. [12]

In addition, Fig. 1 shows that only one direction is taken into account by looking at the orientation of the structures of Attention and RNNN. This means that the calculation must be done in order from t-1 to obtain the $t^{th}$ hidden state, which causes the calculation to slow down. In transformer, these problems were solved by self-attention structure and masking techniques.

The transformer consists of six encoder-decoder structures encoder compresses the input sequence into a single vector expression, and decoder uses this vector expression to create an output sequence in Fig. 2 [4]. Tokenized input data is mapped through the input embedding process to a real-time vector that the computer can understand. In addition, the position information and embedding vectors of each word are added during the positional encoding process. This added value enters the multi-head attention structure and is processed. In decoder, unlike encoder, it consists of two multi-head-attentions, and masking adds the process of masking words that are later than their current location to prevent them from being focused.

PE in the positional encoding process returns the value added by the sin, cos function, and Embedding vector. The cos function is used when the sin function is odd when the index is even. Using the above method, the value of the vector varies depending on the location of the same word.

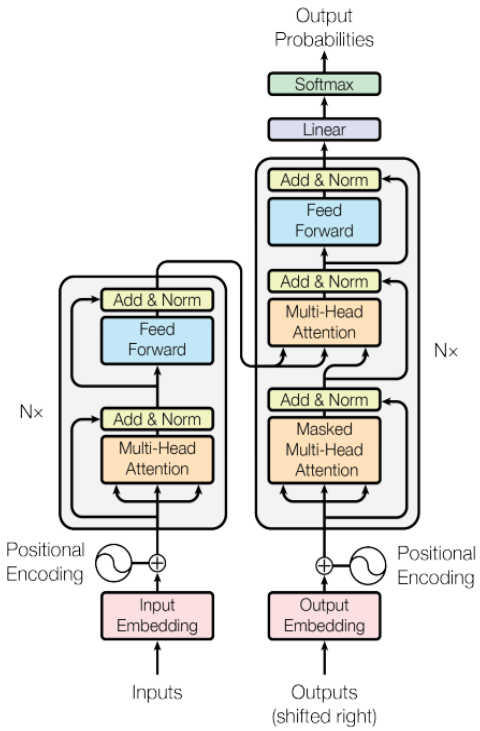
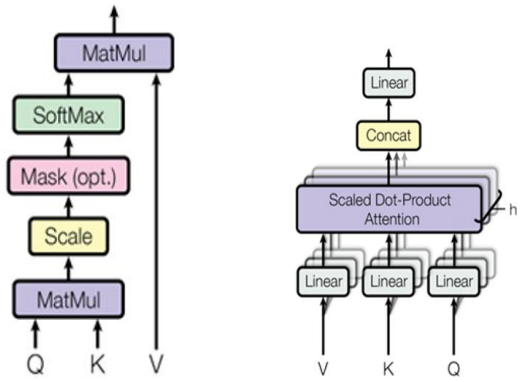



Figure 2. Structure of transformer. [4]

In Fig. 3, multi-head attention is a structure that has eight self-attentions [4]. Self-attention has Q, K, and V vectors with 64 dimensions of Q (Query), K (Key), and V (Value) all of the same values in the existing Attention structure. With Q, K, and V, you can find the association of each word with different sentences.




(a) Scaled dot-product attention (b) Multi-head attention

Figure 3. The structures of attention and multi-head attention. [4]

According to the attenuation structure, the overall formula is as follows, except for the Mask portion used in the Decoder.

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

In the equation (1), the value divided by the $d_k$ (level of the vector) after the matrix is calculated is softmax() to obtain the matrix of Q and K vectors. This represents the probability of where to focus on each word of the sentence.

Multi-Head-Attention consists of eight parallel self-Attention. At this time, each Attention matrix is called Attention Head. And the weighting matrix for Q, K and V at this time is called WQ, WK and WV. This is because each head can be associated with a different perspective when the association with words is obtained from the Q vector.

The data pre-processing process was largely divided into data collection, data refining, and Tokenize processes. First, in data collection, the example of Naver's English dictionary was crawling to create 860,000 Young parallel Copper data. In the data refining stage, special characters, unnecessary characters, etc. were removed, and all English sentences were converted to lower case letters. In Tokenize stage, sentences were largely divided into word units with Ntk, Mecab, and BPE-based Sentencepiece.

## III. Experimental Results

The data refining and Tokenize process in this paper used the Ubuntu environment and the Windows environment in the learning process. In the Ubuntu environment, the Tokenize process was carried out using Jupyter Notebook and Tokenizer Ntk, Meebab, and Sentencepiece in Python language.

In Windows environments, Google's Collaboration was used to create and learn Transformer models using Python language and Tensorflow.

Learning 1, Learning 2, Learning 3, and Learning 4 were measured based on accuracy, and comparative experiments were conducted based on BLEU Score comparison and the results of the sentence translation of test data in Table III.

TABLE III. Learning

| | Learning 1 | Learning 2 | Learning 3 | Learning 4 |
|---|---|---|---|---|
| Epoch | 50 | 50 | 40 | 40 |
| Batch Size | 512 | 512 | 1024 | 1024 |
| Input Vocab Size | 8064 | 7951 | 48990 | 13562 |
| Output Vocab Size | 7969 | 7931 | 49923 | 14941 |
| Data Size | 100,000 | 100,000 | 860,000 | 860,000 |
| Tokenizer | Nltk, Mecab | Sentencepiece | Nltk, Mecab | Sentencepiece |

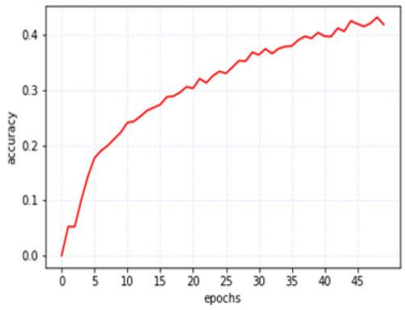



Figure 4. The accuracy of learning 1.

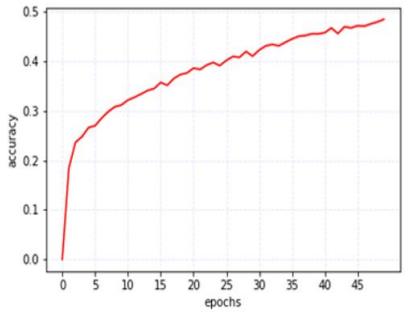



Figure 5. The accuracy of learning 2.

Fig. 4 and Fig. 5 show accuracy of learning 1, learning 2, respectively. Experiments with 100,000 data and fewer vocab sizes showed that learning 2 with BPE-based Tokenizer performed better than learning 1. These results mean that BPE-based Tokenizer is more predictable with fewer vocabs.

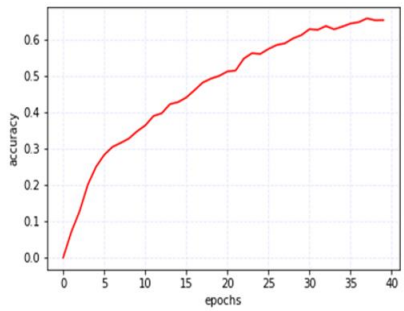



Figure 6. The accuracy of learning 3.

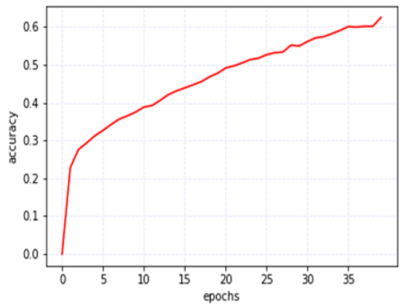



Figure 7. The accuracy of learning 4.

Fig. 6 and Fig. 7 show accuracy of learning 3, learning 4, respectively. More vocabularies were created and learned with 860,000 data, and the performance of the same epoch in Learning 3 was slightly better. Learning 1, learning 2, and learning based on BPE, performed better. These differences appear to have increased the size of the vocabulary as the number of parallel-copper data increases, enabling sufficient prediction of learning 3, which is based on morph-analysis.

TABLE IV. BLEU SCORE FOR LEARNING 3 AND LEARNING 4 AND TIME PER EPOCH (AVERAGE), BLEU SCORE = (SCORE*100)

| | Learning 3 | Learning 4 |
|---|---|---|
| BLEU Score | 28.17 | 24.87 |
| Time taken per epoch | 350 seconds on average | 235 seconds on average |

TABLE V. TRANSLATION RESULTS OF LEARNING 3

| | |
|---|---|
| Sentence1 | Input: casey putsch is the owner of putsch racing |
| | Predicted : 케이시 푸치 는 푸치 레이싱 의 소유주 이 다 |
| Sentence2 | Input : his owner is a man named jon arbuckle |
| | Predicted : 가필드 의 주인 은 존 아버클 이 라는 사람 이 다 |

TABLE VI. TRANSLATION RESULTS OF LEARNING 4

| | |
|---|---|
| Sentence 1 | Input : ▁however , ▁mr s ▁t ' s ▁owner ▁ju de ▁ ry der ▁did ▁not ▁give ▁up ▁hope |
| | Predicted : ▁하지만 ▁주인 은 ▁모 집 ▁주인 은 ▁하지 ▁않았다 |
| Sentence 2 | Input : ▁the ▁user ▁name ▁for ▁the ▁owner ▁is ▁missing |
| | Predicted : ▁소유자 의 ▁사용자 ▁이름이 ▁잘못되었습니다 |

Table IV shows that the time taken per BLEU Score and one epoch of Learning 3, Learning 4. BPE-based Learning 4 showed faster learning than Learning 3, but lower learning scores than Learning 3.

Table V and Table VI show the results of each translation. The translation results show the characteristics of each Tokenizer. Learning 3 shows learning words by dividing them into morphemes and learning 4 by dividing them into words. '_' in learning 4 is to distinguish between writing spares (' ') when dividing words.

In learning 3, you can see learned words translated well. Learning 4 results, on the other hand, show that the results of not sharing words are good, but the results of the translation are worse. English does not lose its meaning even if you divide words into letters. However, the results of learning 4 seem to have been worse because the Korean language changes its meaning when divided into words.

## IV. CONCLUDING REMARKS

Using Transformer, NMT model was created and the results of Korean translation according to Tokenizer were compared. The BPE-based Tokenizer showed small vocabulary sizes and fast learning speed, but the translation results were not good due to its Korean characteristics. The Tokenizer based on morphological analysis showed that regardless of the characteristics of language, it had more parallel-copper data and the larger the size of the vocalic, the higher the performance.

Tokenizer, which fits Korean characteristics, showed better performance based on morphological analysis. However, if the environment is less parallel-copper data and the vocalic size cannot be large, the use of BPE-based Tokenizer will show better performance than the morph-analysis base.

The model was learned based on a translation of Naver's English example. Therefore, the results can be different in other environments such as ordinary conversations, Internet chatting, and newspapers.

## CONFLICT OF INTEREST

The authors declare no conflict of interest.

## AUTHOR CONTRIBUTIONS

The first author, Geumcheol Kim, wrote the program and the manuscript for the paper; the corresponding author, Sang-Hong Lee, conducted the main research design and corrected the manuscript for the paper.

## ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. NRF-2019R1F1A1055423).

## REFERENCES

[1] M. Plappert, C. Mandery, and T. Asfour, "Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks," *Robotics and Autonomous Systems*, vol. 109, pp. 13-26, 2018.

[2] A. Ayadi, A. Samet, F. D. B. D. Beuvron, and C. Zanni-Merk, "Ontology population with deep learning-based NLP: A case study on the biomolecular network ontology," *Procedia Computer Science*, vol. 159, pp. 572-581, 2019.

[3] L. F. Donnelly, R. Grzeszczuk, C. V. Guimaraes, W. Zhang, and G. S. Bisset III, "Using a natural language processing and machine learning algorithm program to analyze inter-radiologist report style variation and compare variation between radiologists when

using highly structured versus more free text reporting," *Current Problems in Diagnostic Radiology*, vol. 48, pp. 524-530, 2019.
[4] A. Vaswani, *et al.*, "Attention is all you need," in *Proc. 31st Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017.
[5] R. A. R. A. Al-Hashemi and S. A. Alsharari, "Instant Arabic translation system for signboard images based on printed character recognition," *International Journal of Machine Learning and Computing*, vol. 3, pp. 384-388, 2013.
[6] L. N. Prskalo and M. B. Bakaric, "The role of homograms in machine translation," *International Journal of Machine Learning and Computing*, vol. 8, pp. 90-97, 2018.
[7] Glue benchmark. [Online]. Available: https://gluebenchmark.com/leaderboard/
[8] F. Abedini, F. Mahmoudi, and A. H. Jadidinejad, "From text to knowledge: Semantic entity extraction using YAGO ontology," *International Journal of Machine Learning and Computing*, vol. 1, pp. 113-119, 2011.
[9] E. Buabin, "Boosted hybrid recurrent neural classifier for text document classification on the reuters news text corpus," *International Journal of Machine Learning and Computing*, vol. 2, pp. 588-592, 2012.
[10] Morphological analysis and POS tagging. [Online]. Available: https://konlpy-ko.readthedocs.io/ko/v0.4.3/morph/
[11] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, vol. 1, pp. 1715-1725.
[12] How does attention work in encoder-decoder recurrent neural networks. [Online]. Available: https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/



**Geumcheol Kim** was born in 1995 in Korea and received the B.S. degree in computer science & engineering from Anyang University, Korea in 2020. He is interested in Java, C#, Android in programming techniques. He is currently working as a computer programmer in software company, Korea. His research focuses on deep learning systems, signal processing, speech recognition systems, and HCI systems.

**Sang-Hong Lee** received the B.S., M.S., and Ph.D. degrees in computer science from Gachon University, Korea in 1999, 2001, and 2012, respectively. He is currently an assistant professor in the department of computer engineering at Anyang University, Korea. His research focuses on deep learning systems, neuro-fuzzy systems, biomedical signal processing systems, and biomedical prediction systems.