KOMBO: Korean Character Representations Based on the Combination Rules of Subcharacters

SungHo Kim[1]*, Juhyeong Park[1]*, Yeachan Kim[1], SangKeun Lee[1,2]
[1]Department of Artificial Intelligence, Korea University, Seoul, South Korea
[2]Department of Computer Science and Engineering, Korea University, Seoul, South Korea
{sungho3268,johnida,yeachan,yalphy}@korea.ac.kr

Abstract

The Korean writing system, *Hangeul*, has a unique character representation rigidly following the invention principles recorded in *Hunminjeongeum*.[1] However, existing pre-trained language models (PLMs) for Korean have overlooked these principles. In this paper, we introduce a novel framework for Korean PLMs called **KOMBO**, which firstly brings the invention principles of *Hangeul* to represent character. Our proposed method, **KOMBO**, exhibits notable experimental proficiency across diverse NLP tasks. In particular, our method outperforms the state-of-the-art Korean PLM by an average of 2.11% in five Korean natural language understanding tasks. Furthermore, extensive experiments demonstrate that our proposed method is suitable for comprehending the linguistic features of the Korean language. Consequently, we shed light on the superiority of using subcharacters over the typical subword-based approach for Korean PLMs. Our code is available at: https://github.com/SungHo3268/KOMBO.

1 Introduction

Determining the representation of a word is the first stepping stone in building pre-trained language models. The predominant approach in English has attempted to decompose each word (e.g., *pureness*) into subwords (e.g., *pure* + *ness*) based on the frequency of the words, such as byte-pair encoding (BPE) (Sennrich et al., 2016), WordPiece (Schuster and Nakajima, 2012), and SentencePiece (Kudo and Richardson, 2018). Notably, even for the Korean language, which has a significantly different linguistic structure from English, the subword-based approach has been widely adopted for Korean PLMs (Park et al., 2020, 2021).

However, the Korean writing system (known as *Hangeul*) has a unique property in representing letters. Unlike English, which typically adheres to a "word-subword-character" structure, Korean includes an additional "subcharacter" level consisting of *chosung* (initial consonants), *joongsung* (vowels), and *jongsung* (final consonants). This leads to a "word-subword-character-subcharacter" structure, giving rise to distinct Korean linguistic features. This structure is even problematic when generating compound words, such as 기찻길$_{\text{train track}}$.[2] Despite these nuances in subcharacters with the same meaning, subword-based methods are blind to this information, typically parsing the compound word as separate tokens: 기, 찻, and 길. Such structural differences raise questions about the suitability of the subword-based method for the Korean PLMs, considering that it may overlook important linguistic information beyond characters.

In this work, given that it is crucial to ground the linguistic nuances and unique structures of the Korean language, we draw our attention towards subcharacter in building Korean PLMs. To accurately reflect the linguistic information of subcharacters, we bring the historical document *Hunminjeongeum*, which provides comprehensive insights into the design principles of subcharacters and their combination rules. The following are the two essential statements related to subcharacters (National Hangeul Museum, 2021):

- **[*Design of the Letters*]** Chosung (initial consonant), joongsung (vowel), and jongsung (final consonant) are combined to form a single character representing a syllable.

- **[*Combination of the Letters*]** Chosung can be placed above joongsung, or it can be positioned to the left of joongsung. Jongsung is placed below chosung and joongsung.

*These authors contributed equally to this work.

[1]*Hunminjeongeum* is a book published in 1446 that describes the principles of invention and usage of *Hangeul*, devised by King Sejong (National Hangeul Museum, 2018).

[2]기찻길$_{\text{train track}}$ is formed from 기차$_{\text{train}}$ and 길$_{\text{track}}$, where a subcharacter 'ㅅ' is inserted between two words.

Building on these principles, we propose a novel framework for Korean PLMs, referred to as KOMBO (**KO**rean character representations based on the co**MB**inati**O**n rules of subcharacters). To instill the design principles and combination rules of subcharacters into PLMs, KOMBO starts with subcharacters as the initial representations. These representations are progressively combined to form a character through a merging layer guided by the combination principles. Additionally, we also introduce a masking strategy tailored to subcharacters, aimed at learning the structural knowledge of characters during pre-training. Consequently, such a comprehensive and fundamental approach enables PLMs to better comprehend the unique structure and composition of the Korean language.

To demonstrate the efficacy of the proposed method, we perform extensive experiments over a wide range of NLP tasks and compare ours with a variety of Korean tokenization methods in building PLMs. The results convincingly show that considering subcharacter, especially for Jamo, in a principled manner brings substantial improvement to Korean PLMs. Moreover, the in-depth analysis supports that KOMBO can better understand the Korean features, such as conjugations and offensive language, compared to existing methods. In summary, the contributions of this paper include the following:

- We present KOMBO, a novel framework for Korean PLMs grounded in the invention principles of *Hangeul* as specified in *Hunminjeongeum*.
- We integrate the design of characters and combination rules into neural language models, marking a novel exploration in Korean PLMs.
- We demonstrate that considering the structure of *Hangeul* through invention principles leads to remarkable performance on a wide range of NLP tasks, thereby shedding light on the potential of Jamo units for Korean PLMs.

2 Related Work

2.1 Korean Pre-trained Language Models

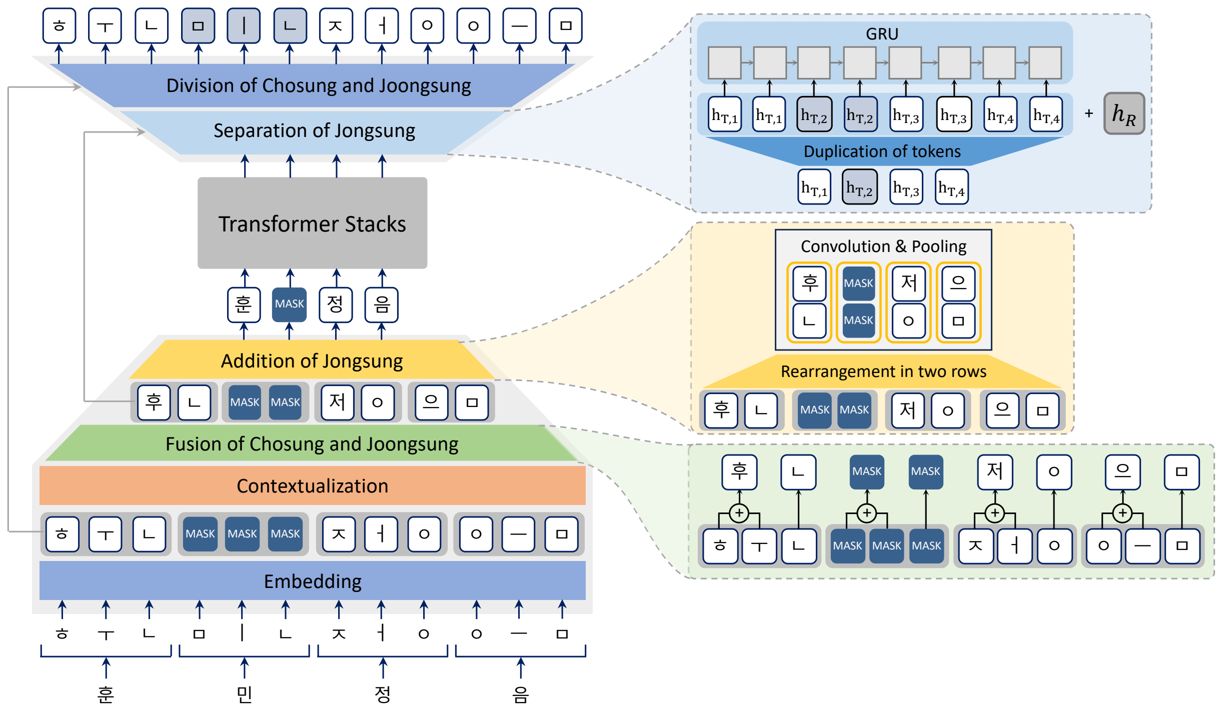
Pre-trained language models have understood the word by splitting it into various atomic units. Similar to other languages, Korean PLMs have also adopted tokenizing sentences into character-level (Cho et al., 2019), subword-level (Lee et al., 2018; Park et al., 2018), or have split by whitespace (Eo et al., 2022). Moreover, to consider the morphologically rich feature of the Korean language, Lee and Shin (2021) and Moon et al. (2022) have utilized a morpheme analyzer (Kudo, 2005) to tokenize sentences. Park et al. (2020), Kim et al. (2021), and Park et al. (2021) have presented morpheme-aware subword tokenization, which is the subword-level tokenization approach applying a morpheme analyzer before implementing BPE. Within this diverse range of units, most of the current Korean PLMs have used morpheme-aware subword units as basic tokens to split the sentence because of their high performance. However, subword-level methodologies have been insufficient in addressing the distinctive linguistic attributes (Albright and Kang, 2009; Park and Shin, 2018) inherent in the Korean language, which are influenced by the presence of subcharacters specific to *Hangeul*.

2.2 Korean Subcharacter Units

To consider the unique structural information of Korean letters, there have been some trials to tokenize the word into subcharacters for word embeddings. Stratos (2017) and Moon and Okazaki (2020) have split words into the subcharacter unit Jamo, which is used as a basic subcharacter of *Hangeul* letters. Additionally, Kim et al. (2022) have decomposed Jamo into smaller tokens, BTS units, which are inspired by the invention principle of *Hangeul*. Depending on the decomposition level, they have discerned BTS units into three different units, such as Stroke, Cji, and BTS. However, most of the previous works have been exploited only in static word embeddings, leaving a research gap in the application in PLMs, such as BERT (Devlin et al., 2019). Therefore, in this work, we propose a novel framework for Korean PLMs based on the design principles and combination rules of subcharacters as mentioned in *Hunminjeongeum*.

3 KOMBO

In this section, we elaborate KOMBO in details for Korean PLMs. The framework begins with taking subcharacters as initial representations (§3.1). These subcharacter representations are progressively combined to form a character based on the combination rules (§3.2). After passing through the transformer blocks (§3.3), the subcharacter representations are reconstructed to perform token-level objectives (§3.4). To learn the structural knowledge of subcharacters during pre-training, we also



ㅎ ㅜ ㄴ ㅁ ㅣ ㄴ ㅈ ㅓ ㅇ ㅇ ㅡ ㅁ
Division of Chosung and Joongsung
Separation of Jongsung
Transformer Stacks
훈 MASK 정 음
Addition of Jongsung
후 ㄴ MASK MASK 저 ㅇ 으 ㅁ
Fusion of Chosung and Joongsung
Contextualization
ㅎ ㅜ ㄴ MASK MASK MASK ㅈ ㅓ ㅇ ㅇ ㅡ ㅁ
Embedding
ㅎ ㅜ ㄴ ㅁ ㅣ ㄴ ㅈ ㅓ ㅇ ㅇ ㅡ ㅁ
훈 민 정 음
GRU
$h_{T,1}$ $h_{T,1}$ $h_{T,2}$ $h_{T,2}$ $h_{T,3}$ $h_{T,3}$ $h_{T,4}$ $h_{T,4}$ + h_R
Duplication of tokens
$h_{T,1}$ $h_{T,2}$ $h_{T,3}$ $h_{T,4}$
Convolution & Pooling
Rearrangement in two rows

Figure 1: Overall illustration of KOMBO where the input is "훈민정음$_{Hunminjeongeum}$" which has four characters and twelve subcharacters. The model starts with the twelve subcharacters and progressively combines them to construct four characters based on the combination principles, e.g., ㅎ+ㅜ+ㄴ → 후+ㄴ → 훈. After going through the transformer stack, the model is trained to predict the consecutive subcharacters with the restoration layers.

introduce the span-subcharacter masking strategy (§3.5). Figure 1 illustrates the overall procedures.

3.1 Initial Representations of KOMBO

KOMBO begins with treating subcharacters as atomic units for representing a word. While there are a number of ways to represent subcharacters (e.g., Stroke, Cji, BTS), we primarily use **Jamo** in the following sections for the sake of simplicity in explanation.[3]

Subcharacter Tokenization Each input character is decomposed into chosung, joongsung, and jongsung. Chosung and joongsung are essential components to create a character, whereas jongsung is an optional component. To represent the absence of jongsung, we use the special empty token (▁), ensuring that each character is always represented with three Jamo units (e.g., a character '차$_{\text{car}}$' is decomposed to ㅊ, ㅏ, and ▁). We denote the subcharacter input sequence as $\mathbf{x} = \{x_1, x_2, \cdots, x_N\} \in \mathbb{R}^N$, where N is the number of subcharacters.

Subcharacter Embedding We project the input sequence $\mathbf{x}$ onto the embedding space. The projected token representations are denoted as follows:

$$\mathbf{e} = \text{Embedding}(\mathbf{x}) \in \mathbb{R}^{N \times D} \qquad (1)$$

where D is the dimension of embeddings.

3.2 Subcharacter Combination

Contextualization Building on the principles of combination, we integrate subcharacter representations to construct a character. However, simply merging these representations without considering context and sequence fails to capture the intricate composition of characters. Therefore, we apply contextualization to the subcharacters using shallow local transformer blocks, subsequently followed by Gated Recurrent Units (GRU) (Cho et al., 2014). Through the contextualization layers, the subcharacter representations are derived as follows:

$$\mathbf{h} = \text{GRU}(\text{LocalTransformer}(\mathbf{e})) \in \mathbb{R}^{N \times D} \qquad (2)$$

Merging Jamo units In the following sections for the intuitive illustration, we refer to the structural representation $h_i \in \mathbf{h}$ as chosung $h_{\text{I},k}$, joongsung $h_{\text{V},k}$, and jongsung $h_{\text{F},k}$, respectively:

$$h_i = \begin{cases} h_{\text{I},k} & \text{if } i = 3k - 2 \\ h_{\text{V},k} & \text{if } i = 3k - 1 \\ h_{\text{F},k} & \text{if } i = 3k \end{cases} \qquad (3)$$

[3]Details regarding BTS units are provided in Appendix A.

where each $k \in [1, N/3]$ corresponds to a character index in the sequence.

Subsequently, to form a character representation, we combine denoted subcharacters in two steps. Depending on the order of combining subcharacters as referred to ***Design of the Letters***, we first merge chosung and joongsung through element-wise summation of $\mathbf{h}_{\mathrm{I}} = \{h_{\mathrm{I},k}\}$ and $\mathbf{h}_{\mathrm{V}} = \{h_{\mathrm{V},k}\}$.

$$\mathbf{h}_{\mathrm{I+V}} = \mathbf{h}_{\mathrm{I}} + \mathbf{h}_{\mathrm{V}} \quad \in \mathbb{R}^{\frac{N}{3} \times D} \tag{4}$$

The second merging step is to combine the resultant representations with jongsung. Considering that jongsung is always positioned below chosung and joongsung, as mentioned in ***Combination of the Letters***, we treat it by performing vertical concatenation of jongsung representations $\mathbf{h}_{\mathrm{F}} = \{h_{\mathrm{F},k}\}$ with the combined chosung and joongsung representations $\mathbf{h}_{\mathrm{I+V}}$. Formally, this can be represented as follows:

$$\mathbf{h}_{\mathrm{R}} = \begin{bmatrix} \mathbf{h}_{\mathrm{I+V}} \\ \mathbf{h}_{\mathrm{F}} \end{bmatrix} \in \mathbb{R}^{2 \times \frac{N}{3} \times D} \tag{5}$$

To generate the final subcharacter jongsung based on the merged representations, we perform convolution and pooling operations[4] over the concatenated representations $\mathbf{h}_{\mathrm{R}}$.

$$\mathbf{h}_{\mathrm{C}} = \mathrm{AvgPool}(\mathrm{Conv}(\mathbf{h}_{\mathrm{R}})) \in \mathbb{R}^{\frac{N}{3} \times D} \tag{6}$$

This results in a dense character representation grounded in the combination rules of subcharacters.

3.3 Transformer Stack

On top of the merged representations, we employ a series of transformer, consisting of L layers, to achieve contextualization as follows:

$$\mathbf{h}'_{\mathrm{C}} = \mathrm{Transformer}_{L}(\mathbf{h}_{\mathrm{C}}) \tag{7}$$

The configuration of these transformer layers follows the same design as if the BERT model (Devlin et al., 2019). Notably, it is crucial to highlight that our framework is primarily designed to manipulate token representations, allowing for seamless integration with various transformer-based architectures, extending beyond BERT.

3.4 Subcharacter Restoration

In sentence-level tasks, the first token (i.e., [CLS]) from the transformer stack is utilized to perform the specified task. However, token-level classification necessitates a fine-grained sequential output that aligns with the vocabulary, requiring subcharacter representations. We thus introduce the restoration layers after the transformer stack, which convert the character representations back into the constituent subcharacters.

The reconstruction proceeds by reversing the process of the subcharacter combination. First, the character representations ($\mathbf{h}'_{\mathrm{C}}$ or $\mathbf{h}'_{\mathrm{R}}$) is duplicated by the number of tokens used in each subcharacter combination process (Eq. (6) or Eq. (4)). Inspired by *U-Net* (Ronneberger et al., 2015), the duplicated representations are subsequently combined with the original subcharacter representations (i.e., $\mathbf{h}_{\mathrm{R}}$ or $\mathbf{h}$) for better restoration. Finally, by leveraging the GRU layer, we ensure the continuity between subcharacters during the reconstruction process. In summary, the restoration process[5] can be formulated as follows:

$$\mathbf{h}'_{\mathrm{R}} = \mathrm{GRU}(\mathrm{Repeat}(\mathbf{h}'_{\mathrm{C}}) + \mathbf{h}_{\mathrm{R}}) \quad \in \mathbb{R}^{\frac{2}{3}N \times D} \tag{8}$$

$$\mathbf{h}' = \mathrm{GRU}(\mathrm{Repeat}(\mathbf{h}'_{\mathrm{R}}) + \mathbf{h}) \quad \in \mathbb{R}^{N \times D} \tag{9}$$

3.5 Span-Subcharacter Masking

Furthermore, to learn the linguistic structure within characters and their corresponding subcharacters, we also introduce a span-subcharacter masking strategy inspired by SpanBERT (Joshi et al., 2020). Specifically, instead of masking tokens at arbitrary positions, we mask out the consecutive three subcharacters (corresponding to chosung, joongsung, and jongsung) of each character for the objective of MLM. Such a masking strategy encourages the model to learn the compositional relationships between subcharacters in the pre-training phase, thereby leading to subcharacter-aware pre-trained language models.

4 Experiments

In this section, we conduct a comparative analysis between the proposed method and existing Korean tokenization models on both standard Korean datasets and noisy Korean datasets.

[4]We heuristically explore and use (2×1) kernels with the stride of one to perform convolution.

[5]Comparison with the variation of restoration process is provided in Appendix B.1.

Model	Tokenization	Vocab Size	KorQuAD	KorNLI		KorSTS		NSMC		PAWS-X	
			Dev(EM/ F1)	Dev	Test	Dev	Test	Dev	Test	Dev	Test
BERT	Stroke	130	38.40/ 50.42	57.78	57.66	67.99	67.62	87.02	86.84	56.49	54.67
	Cji	136	32.64/ 44.48	56.00	57.37	63.86	64.38	86.85	86.70	55.77	54.67
	BTS	112	18.78/ 30.30	50.77	50.52	56.63	60.14	86.14	86.23	54.45	55.33
	Jamo	170	55.73/ 68.90	62.27	64.73	77.22	72.96	87.84	87.78	60.40	59.09
	Character	2K	55.67/ 74.67	72.51	72.59	83.98	76.34	89.03	88.89	69.51	68.70
	Morpheme	32K	65.28/ 80.73	73.90	72.57	82.19	74.35	87.55	87.40	72.97	66.66
	Subword	32K	70.50/ 84.23	73.14	73.32	83.80	76.41	89.03	88.91	72.31	68.88
	MorSubword	32K	66.20/ 80.69	73.91	73.76	84.26	77.29	**89.59**	**89.40**	71.23	68.14
	Word	64K	2.45/ 8.86	66.77	65.45	72.20	65.50	74.91	74.16	65.42	61.16
KOMBO	Stroke	130	64.64/ 75.59	70.31	70.58	82.42	75.55	87.65	87.47	65.25	63.79
	Cji	136	68.84/ 79.68	71.74	72.13	83.37	75.28	86.55	88.08	66.56	66.11
	BTS	112	58.24/ 69.49	66.77	65.97	79.73	72.90	86.68	86.40	62.86	62.49
	Jamo	170	**77.47**/ **86.30**	**74.22**	**73.77**	**84.47**	**77.43**	88.71	88.70	**73.66**	**70.88**

Table 1: Performance of various tokenization methods for PLMs on standard Korean datasets. The evaluation metrics for each task are as follows: KorQuAD: Exact Match/ macro F1, KorNLI: accuracy (%), KorSTS: 100 $\times$ Spearman correlation, NSMC: accuracy (%), PAWS-X: accuracy (%). The best and second-best results are highlighted in **boldface** and underline, respectively.

4.1 Experimental Setup

Models We compare our proposed methods with Korean PLMs using 9 distinct tokenization methods, including BTS units (Stroke, Cji, and BTS), Jamo, Character, Morpheme, Subword, MorSubword (short for Morpheme-aware Subword (Park et al., 2020, 2021; Kim et al., 2021)), and Word units. We uniformly apply BERT to all PLMs, using the same configuration as BERT$_{base}$ with 12 transformer blocks. Depending on the types of subcharacters, we categorize our methods as KOMBO$_{\text{Stroke}}$, KOMBO$_{\text{Cji}}$, KOMBO$_{\text{BTS}}$, and KOMBO$_{\text{Jamo}}$. We match the size of all our models with the state-of-the-art Korean PLM, MorSubword, by adjusting the number of local transformer blocks in the contextualization layer.

Pre-training We pre-trained all models for 1M steps on Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) tasks as BERT. We used the Korean Wiki corpus and the Namuwiki corpus, 6.2 GB, including about 46M sentences in total. The details of data preprocessing and hyperparameter settings are explained in Appendix C.

Evaluation Pre-trained models were individually fine-tuned on each downstream task dataset. We report all experimental results as the average values obtained from three random seeds. To verify the effectiveness of the proposed method in standard Korean dataset, we evaluated the models on five Korean NLU tasks, including machine reading comprehension (KorQuAD 1.0), natural language inference (KorNLI), semantic textual similarity (KorSTS), sentiment analysis (NSMC), and paraphrase identification (PAWS-X). We provide detailed explanations about the overall data information and hyperparameter settings in Appendix D.1.

Moreover, to assess the robustness of models in noisy Korean settings, we synthetically injected the typos into the Korean NLU datasets, such as KorNLI, KorSTS, NSMC, and PAWS-X. We randomly generated typos using four different typo methods, following Zhuang and Zuccon (2021); Insertion: randomly add a letter which is adjacent of the letter on the keyboard, Transposition: randomly switches a letter with one of its neighbor letter, Substitution: randomly changes a letter with one of its neighbor letters on the keyboard, and Deletion: drops a random letter. The other settings, unless specified, follow the same manner as Korean NLU tasks.

4.2 Experiment Results

4.2.1 Standard Korean Datasets

In Table 1, our proposed method KOMBO consistently outperforms BERT models, which use the same subcharacter tokenization as the initial representation, for all tasks with marginal increases,

Model	Tokenization	Clean	Insertion		Transposition		Substitution		Deletion	
			+20%	+40%	+20%	+40%	+20%	+40%	+20%	+40%
BERT	Jamo	64.73	62.87	60.54	56.38	51.31	62.60	61.36	56.47	51.29
	Character	72.59	<u>69.41</u>	<u>66.91</u>	61.59	<u>54.79</u>	<u>68.36</u>	<u>64.57</u>	62.10	54.71
	Subword	73.32	67.22	63.43	62.45	**54.83**	67.17	61.83	62.14	**55.46**
	MorSubword	<u>73.76</u>	68.60	64.63	<u>62.77</u>	54.36	67.62	62.07	<u>62.57</u>	54.86
KOMBO	Jamo	**73.77**	**70.63**	**67.73**	**62.82**	54.54	**70.83**	**68.06**	**63.21**	<u>54.94</u>

Table 2: Performance on KorNLI with typo. We measure the sensitivity to typo rate using four different typo generation methods. The best and second-best results are highlighted in **boldface** and <u>underline</u>, respectively.

averaging more than 9%. We observe that, among ours, KOMBO$_{\text{Jamo}}$ presents the most overwhelming performances, while KOMBO for BTS units shows lagging performances. We suspect that this is due to the truncated initial inputs at subcharacter level by a setting of maximum sequence length, giving rise to short context. KOMBO$_{\text{Jamo}}$ outperforms the state-of-the-art model MorSubword across all tasks except for NSMC. Despite using only 0.53% of the size of the static embeddings compared to the MorSubword model, our method achieves higher performance by an average of 2.11%. These admirable results provide valuable insight into the prospective potential of Jamo, taking over the trend of current subword-based Korean PLMs.

4.2.2 Noisy Korean Datasets

We evaluate the robustness of models to typos in two experimental settings. The first is the random typo setting, where we raise the typos by randomly selecting typo methods among Insertion, Transposition, Substitution, and Deletion, starting from the rate of 0% and increasing it by 5% up to 40%. In Figure 2, the baselines using larger token units, such as Subword and MorSubword, are highly vulnerable to typo, whereas the models using smaller token units, such as Jamo and Character, generally exhibit strong robustness with a gradual decline. Moreover, aligning with the trends shown in the baselines, our proposed model KOMBO$_{\text{Jamo}}$ also exhibited strong robustness to typo. Furthermore, we can observe that the performance gap between KOMBO$_{\text{Jamo}}$ and the state-of-the-art model is getting wider as if the typo rate increases, again demonstrating the strong robustness of our proposed method. The second setting is that we inject only one type of typo error among four different typo methods. In Table 2, we observe that our method is more powerful in the substitution typo method, which is the most similar setting to real-world typographical errors (Jeon et al., 2010). We report more results for other tasks in Appendix D.2.

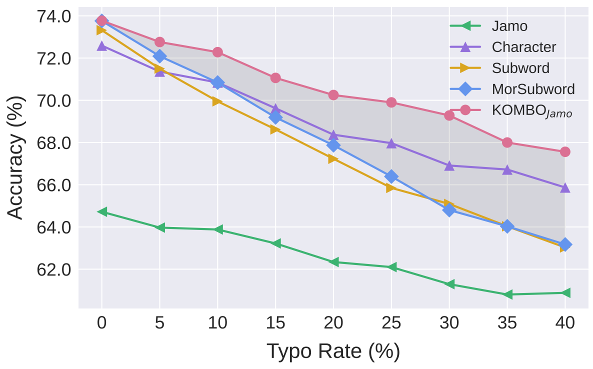


Figure 2: Evaluation results on KorNLI with random typo methods by increasing the typo rate. For better visualization of gap, we fill the area between our proposed method and the state-of-the-art baseline in gray.

5 Ablations

In Table 3, we present ablation experiments on the subcharacter combination methods of KOMBO. The subsequent segments present the experimental results corresponding to the modifications in the structure of subcharacter combination and kernel size used in merging jongsung, respectively. In Table 4, to investigate the effect of Korean character structural knowledge, we compare our subcharacter combination method with other downsampling methods generally used in English.

5.1 Impact of Contextualization

We evaluate our proposed model by removing the contextualization layer before merging embeddings. The omission of the contextualization layer has resulted with a significant drop in performance, which clearly indicates the assisting role of contextualization in character representations. Through these results, we insist that complementing small

Condition	KorNLI		KorSTS		PAWS-X		Total
	Dev	Test	Dev	Test	Dev	Test	Avg
KOMBO$_{\text{Jamo}}$	**74.22**	73.77	**84.47**	**77.43**	**73.66**	**70.88**	**75.74**
Subcharacter Combination							
w/o Contextualization	52.70	54.28	73.11	67.57	54.76	54.77	59.53
w/o Merging Subcharacters	71.45	71.68	82.54	76.27	71.97	68.42	73.72
w/o Addition of Jongsung	73.40	73.69	84.17	75.94	73.17	70.72	75.18
w/o Span-Subcharacter Masking	72.29	73.08	83.58	76.13	72.07	70.21	74.56
Kernel Size in Addition of Jongsung							
w/ (2x2) kernel	73.70	**75.38**	83.04	75.40	65.07	63.13	72.62
w/ (2x3) kernel	73.31	74.68	82.62	75.88	62.73	61.34	71.76
w/ (2x1)+(2x2) kernel	73.78	74.84	82.95	75.27	64.46	62.17	72.25

Table 3: Ablation results for various components of KOMBO$_{\text{Jamo}}$. The first row is the best setting of KOMBO$_{\text{Jamo}}$, which merges all subcharacters sequentially. The best and second-best results are highlighted in **boldface** and underline, respectively.

Method	KorNLI		KorSTS		PAWS-X		Total
	Dev	Test	Dev	Test	Dev	Test	Avg
KOMBO$_{\text{Jamo}}$	**74.22**	**73.77**	**84.47**	**77.43**	**73.66**	**70.88**	**75.74**
Attention Pooling* (Dai et al., 2020)	63.91	63.38	80.58	72.43	54.93	55.37	65.10
Linear Pooling* (Nawrot et al., 2022)	63.93	62.59	80.65	72.02	55.52	55.97	65.11

Table 4: Comparison between KOMBO$_{\text{Jamo}}$ and other downsampling methods used in English (denoted as *). Note that we apply each downsampling methods on KOMBO$_{\text{Jamo}}$ instead of proposed subcharacter combination methods for comparison. The best and second-best results are highlighted in **boldface** and underline, respectively.

embeddings is necessary to enhance the performance of subcharacter models.

5.2 Impact of Span-Subcharacter Masking

Instead of applying the span-subcharacter masking strategy for MLM, we adopt the token-level, i.e., subcharacter masking strategy. We observe that using the span-subcharacter masking strategy enhances the quality of the combined character representation while increasing the average performance by 1%. Through the results, we verify the efficacy of span-subcharacter masking to learn the structural information of characters.

5.3 Impact of Merging Subcharacters

The model that merges subcharacters shows a significant performance improvement, averaging approximately 2.02% higher than the model without subcharacter merging. Furthermore, removing the addition of jongsung results in a performance decrease, averaging around 0.56%. These consistently lower performances underscore the importance of the subcharacter merging steps.

5.4 Impact of Kernel Size

By changing the size and number of the kernels, we explore the most effective kernel size in our subcharacter combination method. The experimental results show that using a (2x1) size kernel, which looks at each character individually, yields the best performance across all tasks except the KorNLI test set. This indicates that the best understanding of Korean is obtained when focusing on the character level, which is the composite unit of Hangeul.

5.5 Impact of Korean Character Structure

In Table 4, to explore the effect of recognition of character structure in Korean, we compare our method with other English downsampling methods (Dai et al., 2020; Nawrot et al., 2022), which do not consider the unique Korean character structure. Instead of our subcharacter combination method,

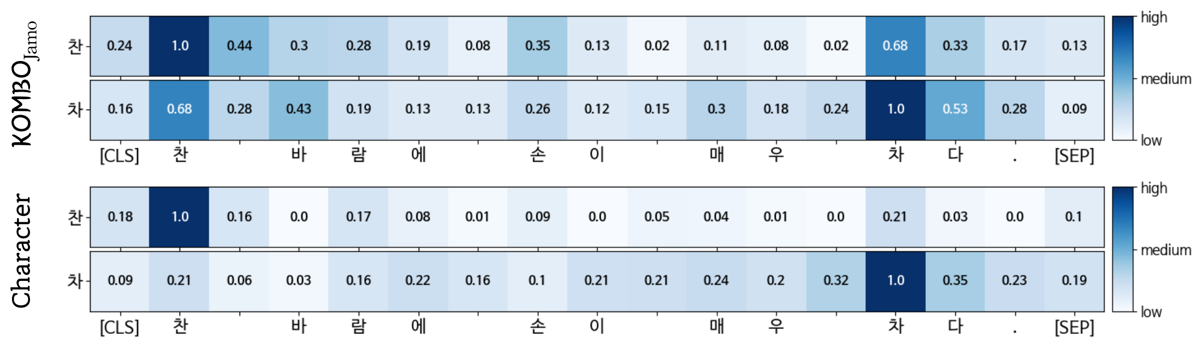


Figure 3: Visualization of the character representations. Given the target sentence as "**찬** 바람에 손이 매우 **차**다. (My hands are very **cold** in the **cold** wind.)", the histogram represents the cosine similarities between each of the two embeddings of the character '찬$_{\text{cold}}$' and '차$_{\text{cold}}$' and all characters in the target sentence.

we opt to employ each downsampling method to combine subcharacter embeddings. Both methods show significant performance drops by an average of over 10% compared to our method. We demonstrate that considering the structure of *Hangeul* is a keystone to comprehending the Korean language.

6 Analysis

In this section, we analyze two Korean linguistic characteristics. The primary objective is to investigate the resilience of predicates against variations arising from Korean conjugations, while the secondary aim involves evaluating the model's robustness to Korean offensive languages.

6.1 Robustness to Character Conjugations

Korean, as an agglutinative language, presents a multitude of conjugational variations occurring at both the character and subcharacter levels through inserting, deletion, or substitution of endings, resulting in alterations in parts-of-speech (POS) or tense. For instance, the verb '차다$_{\text{be cold}}$', composed of the stem '차$_{\text{cold}}$' and the ending '다$_{\text{be}}$', can transform into the adjective '찬$_{\text{cold}}$' by substituting the ending '다$_{\text{be}}$' with the subcharacter 'ㄴ'. Consequently, a profound understanding of the subcharacter structure within *Hangeul* is essential for proficiently handling Korean conjugations. We compare the vanilla character model with KOMBO$_{\text{Jamo}}$, as illustrated in Figure 3. For the given target sentence, we extract embeddings for the characters '찬' and '차' from both models. Subsequently, we compute cosine similarities between these two embeddings and all character embeddings in the target sentence, visualizing the results on a heatmap. The vanilla character model with static embeddings distinguishes '찬$_{\text{cold}}$' and '차$_{\text{cold}}$' as distinct entities, whereas KOMBO$_{\text{Jamo}}$ recognizes them as semantically similar characters, displaying significantly higher similarity scores for both embeddings (0.68) compared to the character model's score of 0.21. Further instances are provided in Appendix E.

6.2 Robustness to Korean Offensive Language

Warning: *This section contains several offensive statements.*

To evaluate the robustness of our proposed method to offensive language, we experiment on the Korean offensive language datasets BEEP! (Moon et al., 2020), K-MHaS (Lee et al., 2022a), and KOLD (Jeong et al., 2022). Details for data and experimental settings are presented in Appendix D.3. In Table 5, KOMBO$_{\text{Jamo}}$ shows the highest macro F1 score across all tasks, demonstrating its robustness on Korean offensive wordings. Interestingly, we find that the subword-based models struggle with the Korean offensive wordings; for example, they tokenize the compound word '개새끼$_{\text{puppy}}$', which is combined '개$_{\text{dog}}$' and '새끼$_{\text{pup}}$', into independent three characters '개', '새', and '끼'. [6]

7 Conclusion

Although there is crucial linguistic information and clear invention principles in Korean letters, the existing Korean PLMs have overlooked them and just opted to use subword-level tokenization methods due to their high performance. In this paper, we first bring attention to the overlooked design principles and combination rules of subcharacters, as specified in *Hunminjeongeum*. We have introduced a novel framework for Korean PLMs called

[6]In Korean, '개새끼$_{\text{puppy}}$' is used as an insult to refer to someone in a disrespectful manner.

Model	Tokenization	BEEP!			K-MHaS			KOLD		
		P	R	F1	P	R	F1	P	R	F1
BERT	Jamo	87.05	69.78	77.43	77.85	73.27	75.13	39.03	47.96	41.95
	Character	<u>89.62</u>	71.27	79.40	<u>78.58</u>	75.24	76.45	<u>46.97</u>	52.82	<u>49.12</u>
	Subword	87.30	71.49	78.61	77.01	75.28	75.90	45.88	<u>53.59</u>	48.66
	MorSubword	89.09	<u>72.45</u>	<u>79.91</u>	77.99	<u>75.63</u>	<u>76.58</u>	46.70	52.45	48.91
KOMBO	Jamo	**90.66**	**74.28**	**81.57**	**79.21**	**76.15**	**77.48**	**47.22**	**55.79**	**50.42**

Table 5: Evaluation results for the robustness of the models on three Korean offensive language datasets. We evaluate the model reporting performance metrics, including Precision (P), Recall (R), and macro F1. The best and second-best results are highlighted in **boldface** and <u>underline</u>, respectively.

KOMBO, which generates character representations following the design and combination rules of subcharacters. We have demonstrated the efficacy of KOMBO on both standard Korean datasets and noisy Korean datasets by outperforming various Korean tokenization baselines on most of the tasks we evaluated. Additionally, through diverse ablations and analyses, we have shown that our proposed method improves the quality of character representations and has a more robust adaptability to Korean conjugations and offensive language. These convincing outcomes of KOMBO$_{\text{Jamo}}$ following the invention principles of *Hangeul* can thereby provide an inspiring insight for the prospective potential of Jamo unit applying in Korean PLMs.

Limitations

While our proposed methodology demonstrates its suitability for Korean natural language processing, it comes with some limitations for future research. Although our proposed KOMBO is designed with an encoder transformer model, our framework is primarily designed to manipulate token representations, allowing for seamless integration with various transformer-based architectures, extending beyond Encoder Models. Therefore, we believe that its dynamic ability to generate appropriate character embeddings for each input also works well in generative modeling, too. We leave the exploration of its value in generative models for future research.

In this work, we limit our focus to Korean representations up to the character-level, as our main objective is to incorporate Korean subcharacter units into PLMs following the principles of *Hangeul*. However, given the morphologically intricate nature of the Korean language, considering representations up to the subword-level is also imperative. With our hierarchical combination approach demonstrating superior performance at the character level, we anticipate that it may serve as a precursor for future methodologies transitioning from character-level to subword-level representations in Korean language processing.

Ethics Statement

We only use three previously collected or synthetically generated Korean offensive language benchmarks, which are annotated with humans who have verified their qualifications. We strictly follow the data usage agreements for each public dataset we implement in this paper. We also mention a warning statement in §6.2, where the offensive statements are directly used in this paper. Additionally, we recognize the potential that the high performance in Korean offensive language also means the model might be biased toward misused or offensive language. However, we believe that the high robustness of our method is aligned with the advantages due to the deep comprehension of Korean character structure, not the familiarity with toxic wording.

Acknowledgements

This work was supported by the Basic Research Program through the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2021R1A2C3010430) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University)).

References

Adam Albright and Yoonjung Kang. 2009. Predicting innovative alternations in korean verb paradigms. *Current issues in unity and diversity of languages: Collection of the papers selected from the CIL 18, held at Korea University in Seoul*, pages 1–20.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Won Ik Cho, Seok Min Kim, and Nam Soo Kim. 2019. Investigating an effective character-level embedding in korean sentence classification. In *Proceedings of the 33rd Pacific Asia Conference on Language, Information and Computation*, pages 10–18, Hakodate, Japan. Waseda Institute for the Study of Language and Information.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Advances in Neural Information Processing Systems*, volume 33, pages 4271–4282, Online. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sugyeong Eo, Chanjun Park, Hyeonseok Moon, Jaehyung Seo, and Heuiseok Lim. 2022. Word-level quality estimation for korean-english neural machine translation. *IEEE Access*, 10:44964–44973.

Jiyeon Ham, Yo Joong Choe, Kyubyong Park, Ilji Choi, and Hyungjoon Soh. 2020. KorNLI and KorSTS: New benchmark datasets for Korean natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 422–430, Online. Association for Computational Linguistics.

Hee-Won Jeon, Daniel Huang, and Hae-Chang Rim. 2010. Analyzing of hangul search query spelling error patterns and developing query spelling correction system based on user logs. In *Annual Conference on Human and Language Technology*, pages 15–21, Gyeongju, Republic of Korea. Human and Language Technology.

Younghun Jeong, Juhyun Oh, Jongwon Lee, Jaimeen Ahn, Jihyung Moon, Sungjoon Park, and Alice Oh. 2022. KOLD: Korean offensive language dataset. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10818–10833, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Boseop Kim, HyoungSeok Kim, Sang-Woo Lee, Gichang Lee, Donghyun Kwak, Jeon Dong Hyeon, Sunghyun Park, Sungju Kim, Seonhoon Kim, Dongpil Seo, Heungsub Lee, Minyoung Jeong, Sungjae Lee, Minsub Kim, Suk Hyun Ko, Seokhun Kim, Taeyong Park, Jinuk Kim, Soyoung Kang, Na-Hyeon Ryu, Kang Min Yoo, Minsuk Chang, Soobin Suh, Sookyo In, Jinseong Park, Kyungduk Kim, Hiun Kim, Jisu Jeong, Yong Goo Yeo, Donghoon Ham, Dongju Park, Min Young Lee, Jaewook Kang, Inho Kang, Jung-Woo Ha, Woomyoung Park, and Nako Sung. 2021. What changes can large-scale language models bring? intensive study on HyperCLOVA: Billions-scale Korean generative pretrained transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3405–3424, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nayeon Kim, Jun-Hyung Park, Joon-Young Choi, Eojin Jeon, Youjin Kang, and SangKeun Lee. 2022. Break it down into BTS: Basic, tiniest subword units for Korean. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7007–7024, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Takumitsu Kudo. 2005. Mecab : Yet another part-of-speech and morphological analyzer.

Chanhee Lee, Dongyub Lee, YunA Hur, Kisu Yang, and Heuiseok Lim. 2018. Comparing byte pair encoding methods for korean. In *Annual Conference on Human and Language Technology*, pages 291–295, Seoul, Republic of Korea. Human and Language Technology.

Jean Lee, Taejun Lim, Heejun Lee, Bogeun Jo, Yangsok Kim, Heegeun Yoon, and Soyeon Caren Han. 2022a. K-MHaS: A multi-label hate speech detection dataset in Korean online news comment. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3530–3538, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Jean Lee, Taejun Lim, Heejun Lee, Bogeun Jo, Yangsok Kim, Heegeun Yoon, and Soyeon Caren Han. 2022b. K-MHaS: A multi-label hate speech detection dataset in Korean online news comment. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3530–3538, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Sangah Lee and Hyopil Shin. 2021. 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)*, pages 410–416, Online. Association for Computational Linguistics.

Seungyoung Lim, Myungji Kim, and Jooyoul Lee. 2019. Korquad1. 0: Korean qa dataset for machine reading comprehension. *arXiv preprint arXiv:1909.07005*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations*, New Orleans, United States.

Jihyung Moon, Won Ik Cho, and Junbum Lee. 2020. BEEP! Korean corpus of online news comments for toxic speech detection. In *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, pages 25–31, Online. Association for Computational Linguistics.

Sangwhan Moon, Won Ik Cho, Hye Joo Han, Naoaki Okazaki, and Nam Soo Kim. 2022. OpenKorPOS: Democratizing Korean tokenization with voting-based open corpus annotation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4975–4983, Marseille, France. European Language Resources Association.

Sangwhan Moon and Naoaki Okazaki. 2020. Jamo pair encoding: Subcharacter representation-based extreme Korean vocabulary compression for efficient subword tokenization. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3490–3497, Marseille, France. European Language Resources Association.

National Hangeul Museum. 2018. A guide to hunminjeongeum. `https://hangeul.go.kr/user/synapView.jsp?filename=BBS/2BD8647E-0CBE-42A9-EEF9-8C5AD8AC70CE.pdf`.

National Hangeul Museum. 2021. Easy reading of hunminjeongeum. `https://hangeul.go.kr/user/synapView.jsp?filename=BBS/A0479188-1C12-D328-AE4D-B4DF9C279181.pdf`.

Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Lukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. 2022. Hierarchical transformers are more efficient language models. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1559–1571, Seattle, United States. Association for Computational Linguistics.

Kyubyong Park, Joohong Lee, Seongbo Jang, and Dawoon Jung. 2020. An empirical study of tokenization strategies for various Korean NLP tasks. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 133–142, Suzhou, China. Association for Computational Linguistics.

Lucy Park. 2016. Naver sentiment movie corpus.

Sungjoon Park, Jeongmin Byun, Sion Baek, Yongseok Cho, and Alice Oh. 2018. Subword-level word vector representations for Korean. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2429–2438, Melbourne, Australia. Association for Computational Linguistics.

Sungjoon Park, Jihyung Moon, Sungdong Kim, Won Ik Cho, Ji Yoon Han, Jangwon Park, Chisung Song, Junseong Kim, Youngsook Song, Taehwan Oh, Joohong Lee, Juhyun Oh, Sungwon Lyu, Younghoon Jeong, Inkwon Lee, Sangwoo Seo, Dongjun Lee, Hyunwoo Kim, Myeonghwa Lee, Seongbo Jang, Seungwon Do, Sunkyoung Kim, Kyungtae Lim, Jongwon Lee, Kyumin Park, Jamin Shin, Seonghyun Kim, Lucy Park, Lucy Park, Alice Oh, Jung-Woo Ha (NAVER AI Lab), Kyunghyun Cho, and Kyunghyun Cho. 2021. KLUE: Korean Language Understanding Evaluation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran.

Suzi Park and Hyopil Shin. 2018. Grapheme-level awareness in word embeddings for morphologically rich languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pages 234–241, Munich, Germany. Springer, Cham.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Karl Stratos. 2017. A sub-character architecture for Korean language processing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 721–726, Copenhagen, Denmark. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3687–3692, Hong Kong, China. Association for Computational Linguistics.

Shengyao Zhuang and Guido Zuccon. 2021. Dealing with typos for BERT-based passage retrieval and ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2836–2842, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Appendix

A KOMBO for BTS units

Kim et al. (2022) introduced Basic, Tiniest Subword (BTS) units for the Korean language, inspired by the invention principle of *Hangeul*. BTS units comprise 5 basic consonants and 3 basic vowels, defined as atomic units in *Hunminjeongeum*. The decomposition of Korean characters into BTS units is referred to as BTS decomposition, where the consonant is split into a maximum of 4 subcharacters (e.g., a consonant 'ㅋ' is decomposed into 'ㄱ' and '-'), and the vowel is split into a maximum of 5 subcharacters (e.g., a vowel 'ㅏ' is decomposed into 'ㅣ' and '·'). Additionally, there are two variant decomposition units: the consonant-only BTS decomposition (denoted as Stroke) and the vowel-only BTS decomposition (denoted as Cji, short for Cheonjiin). In this paper, we consider all applications for BTS units as subcharacter units and present as KOMBO$_{\text{Stroke}}$, KOMBO$_{\text{Cji}}$, and KOMBO$_{\text{BTS}}$. To simplify, we collectively call these three models as KOMBO$_{\text{BTS units}}$. We illustrate KOMBO$_{\text{Jamo}}$ in Figure 4, as a representative example among KOMBO$_{\text{BTS units}}$.

Subcharacter Tokenization Unlike Jamo units, which have three subcharacters for each character, BTS units decompose Jamo into smaller subcharacters. Depending on the type of BTS unit, the number of tokens comprising one character is different. Stroke has 9, Cji has 7, and BTS has 13 tokens for each character. To ensure that the number of tokens forming each Jamo is always maximum, we use the special empty token (▂).

Subcharacter Embedding By the same process in Eq. (1), we project the subcharacter sequence onto the embedding space, denoted as:

$$\mathbf{e} = \{e_1, e_2, \cdots, e_N\} \in \mathbb{R}^{N \times D} \tag{10}$$

where N is the number of subcharacters, and D is the dimension of embeddings.

Contextualization To capture the intricate composition of characters, we apply contextualization to the subcharacters, same as in Eq. (2).

$$\mathbf{h} = \text{GRU}(\text{LocalTransformer}(\mathbf{e})) \in \mathbb{R}^{N \times D} \tag{11}$$

Merging Jamo units The contextualized subcharacter representations are combined to form a character representation similar to §3.2. Before merging chosung, joongsung, and jongsung, there is a preceding step combining the subcharacter representations into the representations of chosung, joongsung, and jongsung. We primarily use Stroke (consonant-only BTS decomposition) in the following sections for the sake of simplicity in explanation. For each $k \in [1, N/9]$, $h_{\text{I},k}$, $h_{\text{V},k}$, and $h_{\text{F},k}$ in Eq. (3) can be denoted as follows:

$$h_{\text{I},k} = \sum_{j=1}^{4} h_{9(k-1)+j} \tag{12}$$

$$h_{\text{V},k} = \sum_{j=5}^{5} h_{9(k-1)+j} \tag{13}$$

$$h_{\text{F},k} = \sum_{j=6}^{9} h_{9(k-1)+j} \tag{14}$$

Then, we sum the combined chosung and joongsung representations to form the intermediate representation $\mathbf{h}_{\text{I+V}}$ with the same process as in Eq. (4).

$$\mathbf{h}_{\text{I+V}} = \mathbf{h}_{\text{I}} + \mathbf{h}_{\text{V}} \quad \in \mathbb{R}^{\frac{N}{9} \times D} \tag{15}$$

To consider the position of jongsung, below chosung and joongsung, we vertically concatenate the combined representations $\mathbf{h}_{\text{I+V}}$ and jongsung $\mathbf{h}_{\text{F}}$, represented in Eq. (5).

$$\mathbf{h}_{\text{R}} = \begin{bmatrix} \mathbf{h}_{\text{I+V}} \\ \mathbf{h}_{\text{F}} \end{bmatrix} \in \mathbb{R}^{2 \times \frac{N}{9} \times D} \tag{16}$$

The resultant representations are then performed using the operations in Eq. (6) to generate the character representation.

$$\mathbf{h}_{\text{C}} = \text{AvgPool}(\text{Conv}(\mathbf{h}_{\text{R}})) \in \mathbb{R}^{\frac{N}{9} \times D} \tag{17}$$

Subcharacter Restoration We also reconstruct the outputs of the transformer stack into subcharacters. Due to the simplicity of the addition of subcharacters into Jamo, we directly decompose intermediate representations into BTS units, jumping the decomposition step from Jamo to BTS units. Thus, the process of subcharacter restoration is almost the same as in Eq. (8) and Eq. (9), but different in the size of the vectors. More formally,

$$\mathbf{h}'_{\text{R}} = \text{GRU}(\text{Repeat}(\mathbf{h}'_{\text{C}}) + \mathbf{h}_{\text{R}}) \quad \in \mathbb{R}^{\frac{2N}{9} \times D} \tag{18}$$

$$\mathbf{h}' = \text{GRU}(\text{Repeat}(\mathbf{h}'_{\text{R}}) + \mathbf{h}) \quad \in \mathbb{R}^{N \times D} \tag{19}$$

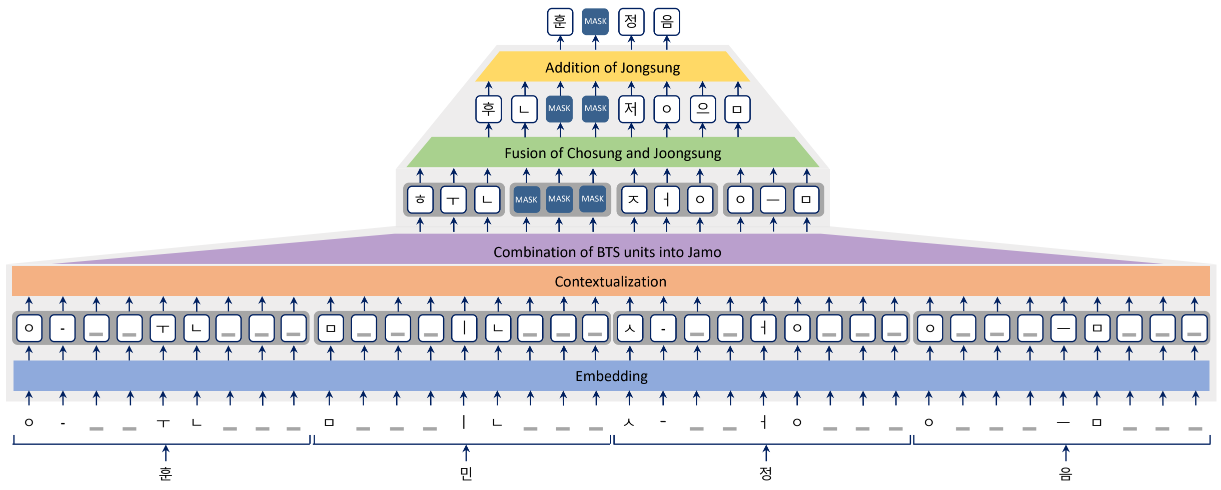


Figure 4: Detailed description of the subcharacter combination method of **KOMBO**$_{\text{Stroke}}$. There is one more merging layer to merge BTS units to Jamo, unlike **KOMBO**$_{\text{Jamo}}$.

Restoration	KorNLI		KorSTS		PAWS-X		Total
	Dev	Test	Dev	Test	Dev	Test	Avg
Linear	73.09$_{\pm 0.77}$	73.96$_{\pm 0.14}$	84.42$_{\pm 0.30}$	76.23$_{\pm 0.45}$	71.51$_{\pm 0.78}$	**71.30**$_{\pm 0.22}$	75.09
Linear+HR	73.69$_{\pm 0.13}$	**74.32**$_{\pm 0.25}$	83.80$_{\pm 0.38}$	76.43$_{\pm 0.62}$	72.80$_{\pm 1.00}$	70.54$_{\pm 0.32}$	75.26
Linear+HR+RC	73.71$_{\pm 0.08}$	74.22$_{\pm 0.24}$	84.15$_{\pm 0.15}$	77.25$_{\pm 0.53}$	72.62$_{\pm 0.69}$	69.08$_{\pm 0.43}$	75.17
GRU	74.04$_{\pm 0.34}$	74.20$_{\pm 0.22}$	83.91$_{\pm 0.23}$	75.91$_{\pm 1.19}$	72.24$_{\pm 1.28}$	68.63$_{\pm 2.00}$	74.82
GRU+HR	73.66$_{\pm 0.70}$	73.88$_{\pm 0.28}$	84.24$_{\pm 0.18}$	76.40$_{\pm 0.42}$	73.04$_{\pm 0.94}$	70.35$_{\pm 0.60}$	75.26
GRU+HR+RC	**74.22**$_{\pm 0.45}$	73.77$_{\pm 0.08}$	**84.47**$_{\pm 0.31}$	**77.43**$_{\pm 0.15}$	**73.66**$_{\pm 0.69}$	70.88$_{\pm 0.58}$	**75.74**

Table 6: Comparison of the variation of restoration process on KorNLI, KorSTS, and PAWS-X tasks. HR means restoring the subcharacter embedding hierarchically, and RC means residual connection. The best and second-best results are highlighted in **boldface** and underline, respectively. Gray area indicates the best Decomposing method.

B Additional Ablations

B.1 Impact of Subcharacter Restoration

We explore the effects of the various subcharacter restoration methods. Table 6 shows the experimental results. We observe that leveraging only a single GRU layer does not help subcharacter reconstruction. However, the GRU layers in hierarchical manners are effective for increasing the granularity of character-level sequences. Moreover, the residual connection, which combines duplicated representations with the original subcharacter embeddings, significantly enhances the performance when combined with the GRU layer, whereas it does not provide any advantages in the case of the linear layer.

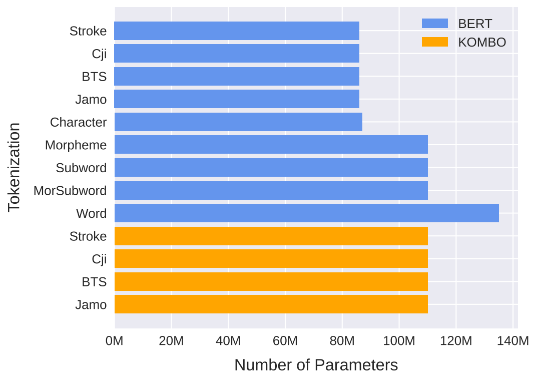
C Details of Pre-training

Data Preprocessing We use the Korean Wiki corpus and the Namuwiki corpus, extracted from dump data using data extractors such as WikiExtractor [7] and NamuWikiExtractor [8]. After applying the data extractor to dump data, we clean the data by removing irregular empty spaces and parsing traces, such as HTML tags. We only retain Korean and English text and punctuation with regular expression. As a result, we can get a total of 6.2 GB, including about 46 million sentences.

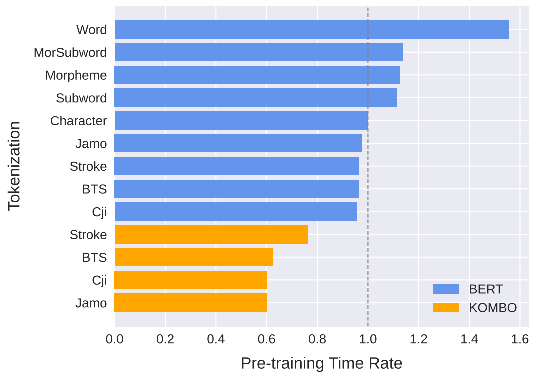
Training Settings Each model is pre-trained for 1M steps with a batch size of 128 using the RTX 3090 GPU. We set the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 5e-05 warm-up over the first 10K steps. We use a sequence length of 128. To match the size of our models with the state-of-the-art Korean pre-trained language model, MorSubword, we adjust the number of local transformer blocks in contextualization layers. The number of parameters in both mod-

[7] WikiExtractor
[8] NamuWikiExtractor

els is 110M, but the training time of KOMBO$_{\text{Jamo}}$ is almost half of vanilla MorSubword model. We compare the number of parameters and the training time rate in Figure 5.



(a)



(b)

Figure 5: Comparison of (a) number of parameters and (b) training time of model according to tokenization method for each model. To facilitate clear quantitative comparison of training times, we express the training time in rate.

D Details of Fine-tuning

D.1 Standard Korean Datasets

Datasets KorQuAD 1.0 (Lim et al., 2019) is the Korean version of the well-known SQuAD 1.0 dataset (Rajpurkar et al., 2016), designed for the machine reading comprehension task. KorQuAD 1.0 dataset consists of 10,645 passages and 66,181 question-answer pairs (60,407 for the training set and 5,774 for the development set). Similar to SQuAD 1.0 dataset, the evaluation involves the model predicting the position of the answer within the given passage.

KorNLI (Ham et al., 2020) is a dataset consisting of 942,854 training sets and 7,500 evaluation sets for natural language inference, sourced from SNLI dataset (Bowman et al., 2015), MNLI dataset (Williams et al., 2018), and XNLI dataset (Conneau et al., 2018). The labels of the dataset are entailment, contradiction, and neutral.

KorSTS (Ham et al., 2020) is designed to evaluate the semantic similarity between two sentences. It is a Korean adaptation of STS-B dataset (Cer et al., 2017). KorSTS dataset comprises 5,749 training examples and 2,879 evaluation examples. Each sample is labeled with a similarity score scaled from 0 to 5, indicating the similarity between the two sentences.

NSMC (Park, 2016) is a NAVER movie review dataset used for performing a sentiment analysis of each Korean sentence. NSMC dataset consists of 150,000 training samples and 50,000 test samples, with each sentence labeled as negative or positive.

PAWS-X (Yang et al., 2019) is a dataset for the paraphrase identification task. PAWS-X dataset has six language tasks, and we only evaluate models on the Korean subset. It consists of 53,338 sentence pairs (49,410 for the training set, 1,965 for the development set, and 1,972 for the test set), and each label has two possible values, different meanings, or paraphrases.

Data Preprocessing To focus solely on evaluating Korean language processing capabilities, we convert English words into *Hangeul* based on the International Phonetic Alphabet (IPA) symbols [9] (e.g., 'bus' → '버스') and remove special characters.

Hyperparameters For each of the datasets, we use the following hyperparameters: KorQuAD 1.0 uses 5 epochs and a batch size of 16 . KorNLI uses 3 epochs and 16 batch size. KorSTS uses 5 epochs and a batch size of 64. NSMC uses 3 epochs and a batch size of 64. PAWS-X uses 5 epochs and a batch size of 64. We select the best learning rate (among 1e-04, 4e-05, and 5e-05) on the Dev set and warmup over the first 10% steps of the total. We use a dropout probability of 0.1. We choose the maximum sequence length (among 128, 256, 512, 1024, and 2048) depending on the type of token.

D.2 Noisy Korean Datasets

We conduct additional typo experiments for the KorSTS, NSMC, and PAWS-X datasets. In Figure 6,

[9] To convert graphemes to phonemes, we leverage the g2pK library. Refer to https://github.com/Kyubyong/g2pK

we illustrate the result of random typo experiments. In Table 7, we present the results of each of the four distinct typo generation methods.

D.3 Korean Offensive Language Dataset

Datasets BEEP! (Moon et al., 2020) is a Korean corpus annotated for toxic speech detection, which consists of 10K manually human-annotated Korean corpus collected from a Korean entertainment news aggregation platform. The labels include Hate, Offensive, and None, categorizing the aggressiveness of the given sentences. To distinguish whether the text contains offensive language or not, we reformulate the labels into 'Hate or Offensive' and 'None' for binary classification.

K-MHaS (Lee et al., 2022b) is a Korean multi-label hate speech detection dataset that consists of 109K utterances from Korean news comments. K-MHaS requires the classification of fine-grained discrimination labels among Politics, Origin, Physical, Age, Gender, Religion, Race, and Profanity based on the input sentences.

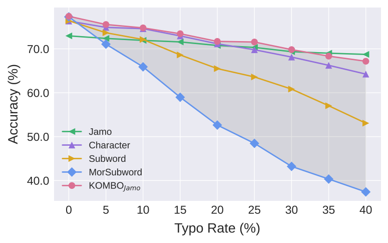
KOLD (Jeong et al., 2022) is a dataset for detecting the Korean offensive language. KOLD comprises 40,429 comments, annotated hierarchically with the type and the target. KOLD has three tasks: Level A (Offensive language detection), Level B (Target types categorization), and Level C (Target group Identification). In this paper, we implement the most difficult task, Level C in KOLD, labeling into 22 target groups: LGBTQ+, Men, Women, Asian, Black, Chinese, Indian, Korean-Chinese, Southeast Asian, White, Conservative, Progressive, Buddhism, Catholic, Christian, Islam, Agism, Disabled, Diseased, Feminist, Physical Appearance, and Socio-economic Status.

For all offensive language datasets, we convert English words into *Hangeul* based on the International Phonetic Alphabet (IPA) symbols.

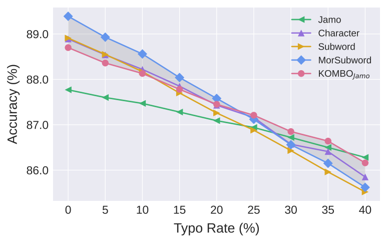
Hyperparameters We train the models on BEEP! for 10 epochs with 32 batch size. In the case of the K-MHaS dataset, we train the models for 4 epochs with 32 batch size. For the experiments on KOLD, we train the models for 5 epochs with a batch size of 32 using a learning rate of 5e-05. We choose the max sequence length (among 128, 256, 512) depending on the type of tokens.

E Robustness to Character Conjugation

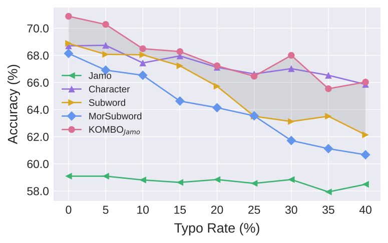
We illustrate more examples about Korean character conjugations in Figure 7, 8, 9, and 10.



(a) PAWS-X



(b) NSMC



(c) PAWS-X

Figure 6: Visualization of the evaluation results on KorSTS, NSMC, and PAWS-X datasets with increasing the typo rate. We fill the gap between our proposed method and the state-of-the-art baseline in gray.

KorSTS										
Model	Tokenization	Clean	Insertion		Transposition		Substitution		Deletion	
			+20%	+40%	+20%	+40%	+20%	+40%	+20%	+40%
BERT	Jamo	72.96	71.67	**69.61**	47.68	29.77	72.15	**69.62**	47.31	30.84
	Character	76.34	70.93	66.33	49.86	32.30	72.34	68.95	49.63	33.81
	Subword	76.41	64.26	57.00	50.52	31.46	68.43	62.64	49.94	32.22
	MorSubword	77.29	66.09	57.26	**57.21**	**34.89**	68.04	57.42	**57.09**	**36.29**
KOMBO	Jamo	**77.42**	**73.13**	69.19	52.34	31.64	**73.21**	68.17	52.47	32.42
NSMC										
Model	Tokenization	Clean	Insertion		Transposition		Substitution		Deletion	
			+20%	+40%	+20%	+40%	+20%	+40%	+20%	+40%
BERT	Jamo	87.78	87.22	**86.58**	84.04	79.17	87.22	86.68	84.08	79.45
	Character	88.89	**87.69**	86.20	84.72	79.44	87.62	86.24	84.95	79.89
	Subword	88.91	87.41	85.86	84.84	79.56	87.31	85.57	84.90	79.73
	MorSubword	**89.40**	87.57	85.72	**85.30**	**79.94**	87.66	85.63	**85.29**	**79.93**
KOMBO	Jamo	88.70	87.57	86.33	84.49	79.15	**87.80**	**86.70**	84.64	79.44
PAWS-X										
Model	Tokenization	Clean	Insertion		Transposition		Substitution		Deletion	
			+20%	+40%	+20%	+40%	+20%	+40%	+20%	+40%
BERT	Jamo	59.09	58.32	57.65	56.95	56.07	58.18	58.84	58.07	56.77
	Character	68.70	66.91	64.91	60.28	**56.98**	66.88	65.33	61.33	57.16
	Subword	68.88	64.63	61.23	60.63	55.40	66.60	62.60	61.37	56.84
	MorSubword	68.14	62.88	60.56	60.25	55.40	63.51	60.67	61.86	**57.65**
KOMBO	Jamo	**70.88**	**69.30**	**67.44**	**63.02**	55.26	**69.09**	**67.44**	**63.51**	57.51

Table 7: Evaluation results on the typo dataset. We measure the sensitivity to typo rate using four different generation methods.The best and second-best results are highlighted in **boldface** and underline, respectively.

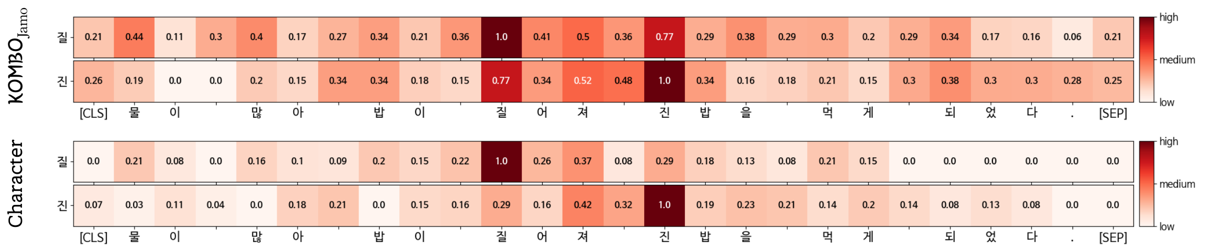


Figure 7: Visualization of the character representations. Given the target sentence as "물이 많아 밥이 **질**어져 **진** 밥을 먹게 되었다. A lot of water makes the rice **mushy**, so I have to eat **mushy** rice.)", the histogram represents the cosine similarities between the embeddings of the character '질' and '진' extracted from the target sentence and all characters in the target sentence.

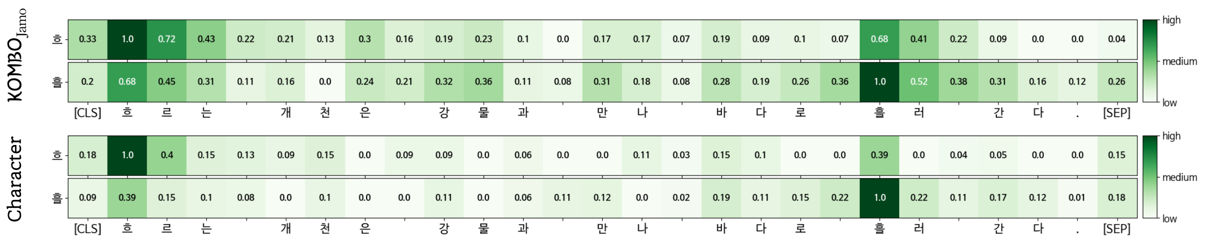


Figure 8: Visualization of the character representations. Given the target sentence as "흐르는 개천은 강물과 만나 바다로 흘러 간다. (A **flow**ing stream joins the river and **flow**s into the sea)", the histogram represents the cosine similarities between the embeddings of the character '흐' and '흘' extracted from the target sentence and all characters in the target sentence.

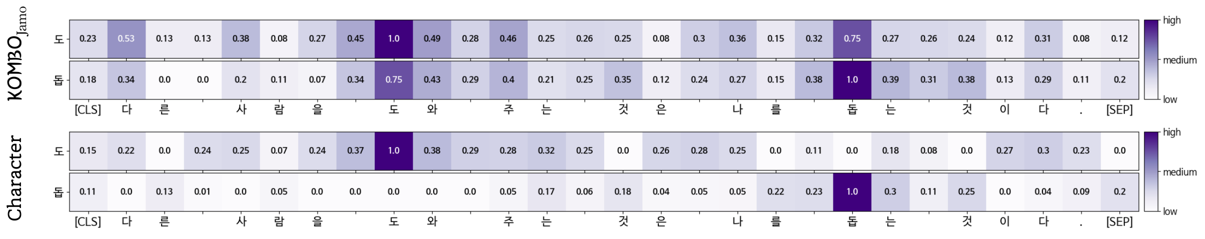


Figure 9: Visualization of the character representations. Given the target sentence as "다른 사람을 도와 주는 것은 나를 돕는 것이다. (**Help**ing others is to **help** myself.)", the histogram represents the cosine similarities between the embeddings of the character '도' and '돕' extracted from the target sentence and all characters in the target sentence.

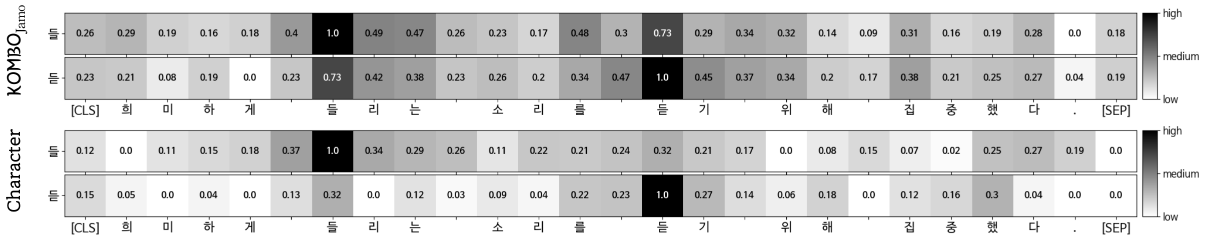


Figure 10: Visualization of the character representations. Given the target sentence as "희미하게 들리는 소리를 듣기 위해 집중했다. (I have to concentrate on **hear**ing the sound, which is unclear to **hear**.)", the histogram represents the cosine similarities between the embeddings of the character '들' and '듣' extracted from the target sentence and all characters in the target sentence.