SCRIPT: A Subcharacter Compositional Representation Injection Module for Korean Pre-Trained Language Models

SungHo Kim[1], Juhyeong Park[1], Eda Atalay[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, edaatalay, yalphy}@korea.ac.kr

Abstract

Korean is a morphologically rich language with a featural writing system in which each character is systematically composed of subcharacter units known as Jamo. These subcharacters not only determine the visual structure of Korean but also encode frequent and linguistically meaningful morphophonological processes. However, most current Korean language models (LMs) are based on subword tokenization schemes, which are not explicitly designed to capture the internal compositional structure of characters. To address this limitation, we propose **SCRIPT**, a model-agnostic module that injects subcharacter compositional knowledge into Korean PLMs. **SCRIPT** allows to enhance subword embeddings with structural granularity, without requiring architectural changes or additional pre-training. As a result, **SCRIPT** enhances all baselines across various Korean natural language understanding (NLU) and generation (NLG) tasks. Moreover, beyond performance gains, detailed linguistic analyses show that **SCRIPT** reshapes the embedding space in a way that better captures grammatical regularities and semantically cohesive variations. Our code is available at https://github.com/SungHo3268/SCRIPT.

1 Introduction

In human writing systems, the grapheme, the smallest unit of written language that encodes linguistic information, plays a crucial role in shaping how meaning is represented and processed (Coulmas, 2003; Sampson, 2015; Daniels and Bright, 1996). In many alphabetic systems, such as English, graphemes typically correspond to atomic letters (e.g., a, b, c), and words are formed through linear combination (e.g., "cat" consists of c, a, and t). However, not all alphabetic systems operate in such a linear manner, nor do they necessarily treat characters as minimal units of written composition.

Korean, in particular, employs a unique featural writing system, *Hangul*, in which each character is a structured composition of smaller subcharacter units known as Jamo. As illustrated in Figure 1(a), each character consists of three *Jamo* units: Choseong (initial consonant), Jungseong (vowel), and Jongseong (final consonant), following fixed spatial arrangements and a strict compositional order. These principles were explicitly defined in *Hunminjeongeum*[1] (National Hangeul Museum, 2018, 2021), the historical document detailing the invention principles of *Hangul*, including the design and combination rules for the subcharacters.

Crucially, this compositional structure is not merely orthographic. As a morphologically rich and agglutinative language, Korean exhibits extensive morphophonological alternations across morpheme boundaries (Lee and Chung, 2003; Matteson et al., 2018; Jun, 2018). Predicate inflection, for example, often triggers systematic subcharacter-level alternations, such as the addition of the final consonant '-ㅆ' to mark past tense or '-ㅁ' for nominalization, as shown in Figure 1(b). Additionally, phonological assimilation between adjacent syllables frequently alters subcharacters to facilitate natural pronunciation (Sohn, 2001; Shin et al., 2012). These phenomena highlight that subcharacter-level features in Korean are tightly linked to grammatical, semantic, and morphophonological functions (Lee and Ramsey, 2001).

Despite this linguistic reality, most contemporary Korean PLMs, including advanced off-the-shelf LLMs (Yoo et al., 2024; LG AI Research et al., 2024), rely almost exclusively on subword-based tokenization. While subword modeling effectively captures lexical semantics from large corpora, it struggles to reflect *Hangul*'s compositional structure, limiting sensitivity to fine-grained morphosyntactic variations (Albright and Kang, 2009; Kim et al., 2025). In contrast, a few subcharacter-

[1]*Hunminjeongeum* explains the letter design and well-formed combinations; modern encoding schemes and keyboard input sequences arise from contemporary standards.

Findings of the Association for Computational Linguistics: ACL 2026, pages 2185–2213

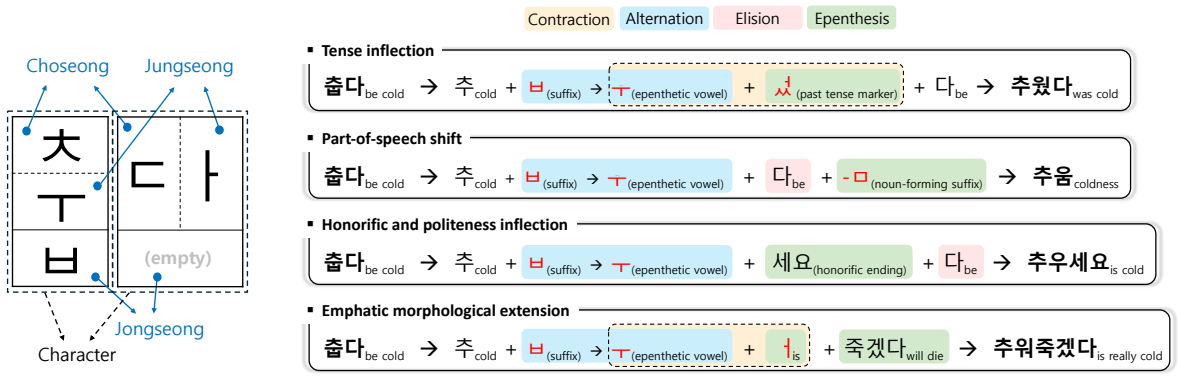


Figure 1: (a) Examples of the components of *Hangul*. This figure illustrates two characters, such as '춥$_{\text{cold}}$' and '다$_{\text{ending suffix}}$', with each subcharacter highlighted in blue. (b) Examples of linguistic phenomena arising from the inflection of predicate '춥다$_{\text{be cold}}$' at the subcharacter-level, with the transformed subcharacters highlighted in red.

based LMs (Moon and Okazaki, 2020; Cognetta et al., 2023; Kim et al., 2024) show strong robustness to such variations but often underperform on downstream tasks due to weaker semantic representations and increased computational cost.

To leverage complementary strengths, we propose **SCRIPT**, a lightweight, plug-and-play module that injects subcharacter-level structural knowledge directly into existing subword-based PLMs. **SCRIPT** attaches to the embedding layer of a PLM and operates through a dual-channel strategy. It compresses subcharacter sequences into structure-aware subword representations grounded in *Hangul*'s compositional principles, and then fuses them with the PLM's original subword embeddings. This design enables the model to capture fine-grained subcharacter compositionality while preserving the rich semantic information learned from large-scale corpora, without modifying the PLM architecture or requiring additional pre-training. The main contributions of this work are summarized as follows:

- We empirically show that most Korean morphological variations occur at the subcharacter-level, motivating subcharacter-aware modeling.
- We introduce **SCRIPT**, a model-agnostic module that injects structure-aware subcharacter compositional representations into existing PLMs via embedding-level integration.
- We show that **SCRIPT** improves performance across a wide range of Korean NLU and NLG benchmarks, while effectively capturing key morphosyntactic phenomena.

2 Motivation

In this section, we present our empirical observations on the pervasiveness of diverse morphophonological changes at the subcharacter-level in real Korean usage, highlighting the importance of modeling the subcharacter structure of *Hangul*.

2.1 Setup

Specifically, we conduct a large-scale corpus-based analysis quantifying their frequency. We used the Korean Part-of-Speech Tagged Corpus[2], which contains 3M words annotated with morpheme-level and POS information. This corpus provides a solid basis for estimating the frequency of subcharacter-level alternations in real usage.

To systematically capture these alternations, we adopted a simple annotation scheme, following Matteson et al. (2018), which marks how each character relates to its base form (lemma). Each character is assigned to one of three categories:

- `KEEP`: unchanged with respect to the base form.
- `MOD`: modified from the base form.
- `NOOP`: omitted in the base form.

For example, in '했다$_{\text{did}}$', whose base form is '하다$_{\text{do}}$', the segment '했$_{\text{did}}$' is labeled as `MOD` because it reflects a tense change arising from the combination of the verb stem '하' and the past tense marker '-었-', which undergoes phonological contraction ('하 + 었 → 했'). In contrast, '다' is labeled as `KEEP` since it remains unchanged.

[2]Part-of-Speech Tagged Corpus (v1.1) from ModuCorpus, provided by the National Institute of Korean Language (2020)

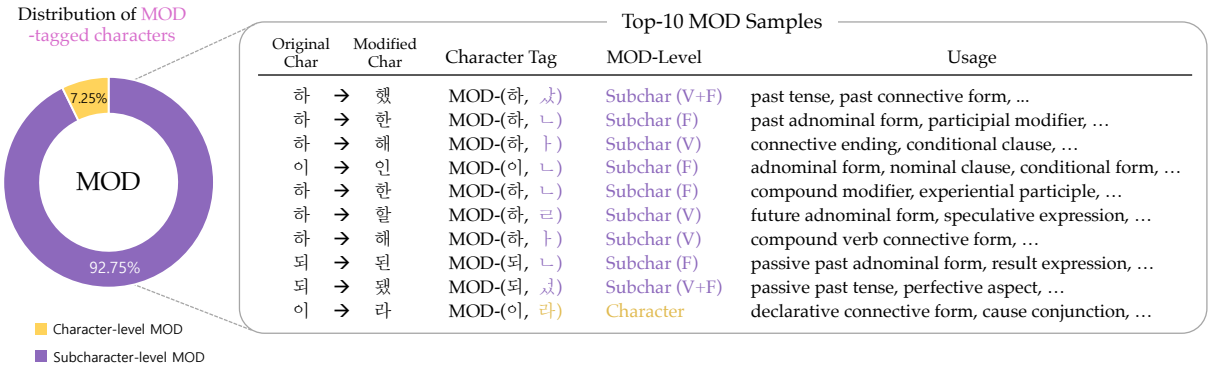


Figure 2: Morphological modifications in a large-scale Korean POS-tagged corpus: the left panel distinguishes subcharacter- and character-level MOD cases, and the right lists the ten most frequent MOD types with raw target characters and corresponding factors.

In this paper, we focus our analysis on characters tagged as MOD, as they directly encode morphological alternations. Each MOD character is further classified by its level of granularity [3]:

- **Subcharacter-level MOD**: when only part of a character is altered. (e.g., the change from '하' to '한', adding the final consonant 'ㄴ').
- **Character-level MOD**: when the entire character is changed into a different character. (e.g., '이' is replaced with '라').

2.2 Observation

As shown in Figure 2, the overwhelming majority of characters tagged as MOD (92.75%) involved subcharacter-level modifications, while only a small fraction (7.25%) represented character-level changes. These findings underscore the importance of modeling subcharacter-level alternations for a deeper and more comprehensive understanding of Korean, especially in adapting to diverse usage and morphological variation. This aligns with prior work (Kim et al., 2024; Lee et al., 2025), which observed that jamo-based language modeling demonstrates robustness in handling character-level conjugation changes and exhibits strong performance on noisy, real-world data such as offensive content. Motivated by this, we aim to explicitly encode subcharacter compositional knowledge in a principled manner, thereby incorporating this linguistic information into language models that have previously overlooked it.

3 Methodology

Based on our observation (§2), we propose **SCRIPT** (**S**ubcharacter **C**ompositional **R**epresentation **I**njection Module for Korean **Pr**e-**T**rained Language Model), a module that enhances PLM's embeddings with subcharacter compositional knowledge. In this section, we instance Jamo as the subcharacter unit in **SCRIPT** and apply it to subword-based PLM, aligning with standard practices in modern Korean PLMs. We also provide extensions for alternative subcharacter units, such as BTS units (Appendix D).

3.1 Overall Framework

SCRIPT is attached to PLMs at the embedding layer, as illustrated in Figure 3(a). Given a Korean text, the model employs two parallel tokenization paths: (1) a subword tokenizer that produces the PLM's original subword sequence, and (2) a subcharacter tokenizer that generates fine-grained input for **SCRIPT**. The subword sequence is projected through the PLM's original embedding layer, while **SCRIPT** constructs an alternative subword-level representation by compressing the subcharacter sequence (§3.2). These two subword representations are then integrated into a unified subword representation (§3.3). This dual-channel strategy allows us to leverage the strengths of both approaches. The full algorithm for synthesizing subword representations with **SCRIPT** is provided in Table 6.

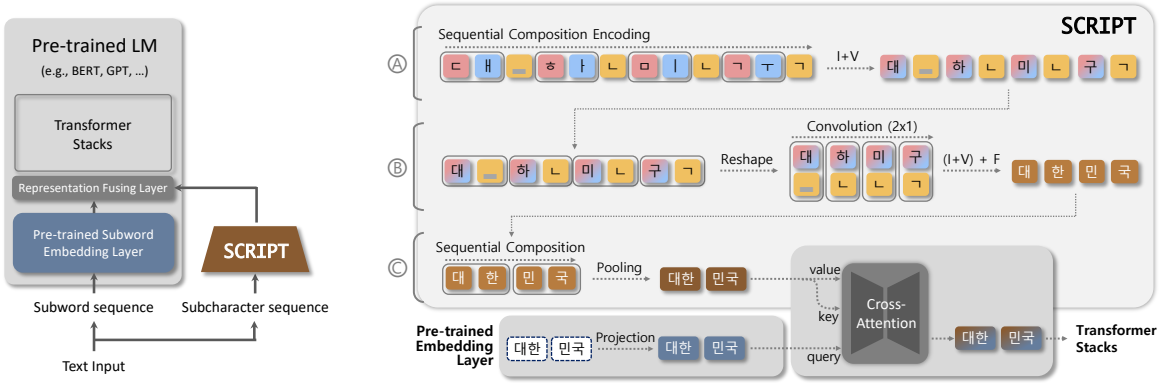
3.2 SCRIPT

Figure 3(b) illustrates the detailed architecture of **SCRIPT**, which compresses subcharacter representations into subword representations in two stages.

3.2.1 Stage 1: Subcharacter-to-Character

The first stage of deriving subword representations from subcharacter representations is to compress subcharacter representations into character repre-

[3] Detailed tagging procedures are provided in Appendix B



(a) (b)

Figure 3: (a) Overall illustration of the PLM enhanced with **SCRIPT**. (b) Detailed architectural example of the **SCRIPT**, starting from the word '대한민국South Korea', which is tokenized into 12 Jamo units: I: [ㄷ, ㅎ, ㅁ, ㄱ], V: [ㅐ, ㅏ, ㅣ, ㅜ], F: [_ , ㄴ, ㄴ, ㄱ]. Each sub-process (A-C) represents a successive fusion step: (A) fusion of Choseong and Jungseong (§3.2.1), (B) addition of Jongseong (§3.2.1), and (C) character-to-subword (§3.2.2).

sentations. Inspired by Kim et al. (2024), to effectively model the compositional structure of *Hangul*, we explicitly incorporate three fundamental compositional principles into our methodology (National Hangeul Museum, 2021; Yeon and Brown, 2013; Unicode Consortium, 2019):

1. ***Composition***: A character is composed of up to three Jamo: Choseong and Jungseong are essential components, whereas Jongseong is not mandatory. [4]
2. ***Spatial arrangement***: Within a syllable block, Choseong is placed either above or to the left of Jungseong, while Jongseong, if present, is always positioned beneath them.
3. ***Sequential order***: Jamo consistently follow a prescribed order: Choseong → Jungseong → Jongseong.

SCRIPT adopts a hierarchical compression architecture grounded in the design principles of *Hangul*, to better capture linguistic features of Korean. This step underpins the 'structure-aware' subword embeddings in **SCRIPT**, as it explicitly encodes the subcharacter compositional knowledge.

Subcharacter Representation. Given an input text $\mathbf{s}$, each character is first decomposed into sequential three subcharacters, I (short for initial consonant, Choseong), V (short for vowel, Jungseong), and F (short for final consonant, Jongseong), following *Principles 1*. If a character lacks a final consonant, a special empty token (_) is inserted in its place. The resulting subcharacter embeddings are denoted as $\mathbf{e} \in \mathbb{R}^{N \times D}$, where N is the number of subcharacter tokens and D is the embedding dimension. Furthermore, to clarify the ordered arrangement of I, V, and F, we denote the sequential subcharacter representation $e_i \in \mathbf{e}$ as $e_{\mathrm{I},k}$, $e_{\mathrm{V},k}$, and $e_{\mathrm{F},k}$ for each integer k in [1, N/3]:

$$e_i = \begin{cases} e_{\mathrm{I},k} & \text{if } i = 3k-2 \\ e_{\mathrm{V},k} & \text{if } i = 3k-1 \\ e_{\mathrm{F},k} & \text{if } i = 3k \end{cases} \quad (1)$$

Fusion of Choseong and Jungseong. To accurately reflect the sequential order (*Principle 3*), we follow the fixed composition sequence (I → V → F). The entire subcharacter sequence is first encoded using a GRU-based sequential composition layer. Following this order, the I and V components are merged via element-wise summation to obtain a combined representation, $\mathbf{h}_{\mathrm{I+V}} \in \mathbb{R}^{\frac{N}{3} \times D}$.

Addition of Jongseong. After forming the intermediate representation of I and V, we incorporate the V to complete the character representation. Reflecting the visual structure of *Hangul*, where F is always positioned below I and V (*Principle 2*), we model this arrangement by vertically concatenating the F representation, $\mathbf{h}_{\mathrm{F}} = \{h_{\mathrm{F},k}\}$, with $\mathbf{h}_{\mathrm{I+V}}$. This composition is formally expressed as follows:

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

To merge these vertically aligned subcharacters into characters, we apply a convolutional layer capturing the relative positional information. We then

[4] A detailed explanation is provided in Appendix A

finalize the character representations by applying average pooling over $\mathbf{h}_R$, yielding dense character representations, $\mathbf{h}_C \in \mathbb{R}^{\frac{N}{3} \times D}$, grounded in the compositional principles of *Hangul*.

3.2.2 Stage 2: Character-to-Subword

The second main stage of **SCRIPT** compresses character representations into subword representations, aligning their granularity with that of the original subwords used in PLMs. Our goal was to aggregate character representations within each subword to form a unified subword representation. However, directly averaging or summing these character representations often led to unstable training. To mitigate this issue, and in line with *Principle 3*, we apply the sequential composition layer once more to capture the compositional order of characters within each subword. Then, we apply a simple pooling operation, specifically, selecting the final character representation at each subword boundary, to obtain the subword representation:

$$\mathbf{h}_S = \text{POOLING}(\text{GRU}(\mathbf{h}_C)) \quad \in \mathbb{R}^{N' \times D} \tag{3}$$

where N' denotes the subword sequence length.

3.3 Fusion of Two Subword Representations

Despite these structured, dense subcharacter-level linguistic features, the resulting subword representations, compressed from subcharacters alone, lack semantic expressiveness, as they are not pre-trained on large-scale Korean corpora. To address this limitation, we fuse them with semantically richer subword embeddings obtained from the existing PLM. Specifically, we introduce a fusion mechanism that integrates two complementary representations: the synthesized subword representation from **SCRIPT**, denoted as $\mathbf{h}_S$, and the original pre-trained subword embedding, $\mathbf{e}_S \in \mathbb{R}^{N' \times D}$, projected into the same embedding space. A cross-attention layer is employed to combine these sources, yielding the final structure-aware subword representation $\mathbf{e}_F \in \mathbb{R}^{N' \times D}$, which is then used as input to the subsequent Transformer layers:

$$\mathbf{e}_F = \text{CROSSATTN}(\text{Q} = \mathbf{e}_S, \text{KV} = \mathbf{h}_S) \tag{4}$$

Through **SCRIPT**, we construct a fused subword representation that integrates the compositional knowledge of *Hangul* with the semantic richness of pre-trained subword embeddings. This dual-channel encoding enhances the language model's ability to capture Korean character structure while preserving subword-level semantic content. These fused representations are then fed into the PLM's Transformer stacks, allowing downstream tasks to benefit from this linguistically enriched input.

4 Experiments

In this section, we evaluate **SCRIPT** on a range of Korean NLU and NLG tasks across strong PLMs (§4.2). We further conduct ablation studies to analyze the contribution of *Hangul*-specific structural knowledge and key design choices (§4.3).

Additionally, we show that this efficiency comes with minimal computational overhead, which remains comparable to standard subword-based models (see Appendix J).

4.1 Experimental Settings

Baselines. We applied **SCRIPT** to four Korean subword-based PLMs ($\text{KoGPT2}_{\text{base}}$, KoGPT3-1.2B, EXAONE-3.5-2.4B-Instruct, $\text{BERT}_{\text{base}}$), and additionally compare with a state-of-the-art Jamo-based encoder model, $\text{KOMBO}_{\text{base}}$ (Kim et al., 2024). Detailed specifications and implementational details are provided in Appendix C, E.

Tasks. We evaluate **SCRIPT**-enhanced models on nine Korean NLU tasks, including four standard benchmarks (KorNLI, KorSTS, NSMC, PAWS-X) and five KoBEST tasks designed to assess diverse linguistic and cognitive capabilities. To evaluate generative performance, we additionally consider KoCommonGen for commonsense reasoning, XL-Sum for summarization, and Korean GEC for grammatical error correction. Detailed dataset statistics and explanations are provided in Appendix E.3.

4.2 Experimental Results

Korean Standard NLU Tasks. As shown in Table 1, **SCRIPT** improves performance across all baselines, yielding average gains of up to 1.6%p. Compared to the Jamo-based baseline $\text{KOMBO}_{\text{base}}$, our $\text{BERT}_{\text{base}}$+**SCRIPT** model achieves superior performance despite using the same underlying architecture and a comparable model size. Unlike $\text{KOMBO}_{\text{base}}$, which directly processes raw subcharacters and relies on costly full pre-training, **SCRIPT** is applied as a plug-in module during only fine-tuning, enabling efficient incorporation of *Hangul* structure. Notably, **SCRIPT** is applicable to both encoder and decoder architectures, overall improving performance across model types.

Model	KorNLI	KorSTS	NSMC	PAWS-X	KoBEST				
					BoolQ	COPA	WiC	HellaSwag	SentiNeg
$\text{KOMBO}_{\text{base}}$	75.97	77.28	88.34	73.40	61.40	61.00	68.91	63.80	79.07
$\text{BERT}_{\text{base}}$	75.85	76.72	**88.96**	72.38	60.75	60.90	73.14	63.20	83.12
$\text{BERT}_{\text{base}}$ + **SCRIPT**	**76.49**	**77.68**	**88.96**	**73.68**	**62.32**	**61.30**	**74.30**	**64.40**	**83.38**
$\text{KoGPT2}_{\text{base}}$	72.24	73.82	**88.90**	76.33	67.22	68.90	67.07	69.10	88.50
$\text{KoGPT2}_{\text{base}}$ + **SCRIPT**	**72.47**	**74.27**	88.80	**76.61**	**68.28**	**70.90**	**68.18**	**72.40**	**89.47**
KoGPT3-1.2B	80.11	76.14	90.51	77.40	77.32	**82.80**	72.78	78.90	96.31
KoGPT3-1.2B + **SCRIPT**	**80.39**	**79.60**	**90.53**	**79.95**	**77.63**	**82.80**	**74.65**	**79.30**	**96.48**
EXAONE-2.4B	83.99	85.08	90.04	85.24	92.59	**93.30**	82.14	85.60	94.21
EXAONE-2.4B + **SCRIPT**	**85.77**	**85.27**	**90.89**	**85.90**	**93.30**	**93.30**	**82.46**	**86.00**	**94.96**

Table 1: Performance on nine Korean NLU tasks. The evaluation metrics for each task are as follows: KorSTS is evaluated using Spearman correlation ×100, while other tasks are evaluated based on accuracy (%). The best results in each family of models are highlighted in **boldface**.

Model	KoCommonGen						
	BLEU 3	BLEU 4	ROUGE-2	ROUGE-L	METEOR	mBERTScore	KoBERTScore
$\text{KoGPT2}_{\text{base}}$	18.29	10.33	44.24	54.50	40.05	83.37	91.21
$\text{KoGPT2}_{\text{base}}$ + **SCRIPT**	**25.01**	**15.57**	**47.42**	**60.00**	**42.53**	**84.67**	**91.46**
KoGPT3-1.2B	26.19	17.20	58.85	62.53	52.11	85.41	91.17
KoGPT3-1.2B + **SCRIPT**	**28.89**	**19.58**	**59.28**	**64.80**	**52.37**	**86.26**	**91.78**
EXAONE-2.4B	40.11	28.41	62.25	64.84	54.84	87.65	93.12
EXAONE-2.4B + **SCRIPT**	**41.48**	**31.80**	**71.03**	**72.16**	**61.27**	**88.12**	**93.95**

Table 2: Performance on KoCommonGen generative task. We use eight automatic evaluation metrics, including n-gram based measures like BLEU, ROUGE, and METEOR, and two BERT-based scores for semantic similarity. The best results in each family of models are highlighted in **boldface**.

Korean Advanced NLU Tasks. **SCRIPT** also outperforms baselines on knowledge-intensive tasks in KoBEST, including reading comprehension (KB-WiC) and commonsense reasoning (KB-HellaSwag). It proves more effective than $\text{KOMBO}_{\text{base}}$, which lags on complex tasks due to its reliance on subcharacter-only inputs. By integrating subword and subcharacter information, **SCRIPT** offers robust and architecture-agnostic enhancements across task complexities.

Korean Generation Tasks. Our method also delivers consistent gains on Korean generative tasks. As shown in Table 2, on KoCommonGen, a task that involves transforming and combining given morphemes to generate plausible sentences, **SCRIPT** improves across all seven generative metrics, with gains (an average of 1.4-3.5%p) depending on model size. The improvements are particularly pronounced in n-gram metrics such as BLEU, METEOR, and ROUGE, indicating enhanced modeling of local compositional patterns in morphologically rich Korean. This trend extends to other generation tasks as shown in Appendix F. Notably, on the Korean GEC task Kor-Learner, **SCRIPT** exceeds the best-performing baseline by over 3.2%p on average. According to Yoon et al. (2023), Kor-Learner includes a high concentration of errors involving particles, endings, and conjugations compared to Kor-Native.

One possible explanation for the relatively larger gains on generative tasks, compared to NLU tasks, is that Hangul's sequential compositional structure (Choseong → Jungseong → Jongseong) aligns naturally with token-by-token decoding, allowing subcharacter information to more directly influence generation decisions. We consider this a promising direction for further analysis, as a deeper understanding of this phenomenon requires further investigation.

4.3 Ablation Study for SCRIPT Architecture

Table 3 presents an ablation study examining the core design choices of **SCRIPT**.

Alternative Tokenization Methods for SCRIPT. Across different granularities, Jamo-based **SCRIPT** achieves the best overall performance. In contrast,

SCRIPT		Fusion w/ PLM	KoBEST					Avg.
Initial Token Unit	Compression		BoolQ	COPA	WiC	HellaSwag	SentiNeg	
Jamo	Principles	CrossAttention	68.28	**70.90**	**68.18**	**72.40**	**89.47**	**73.85**
Stroke	Principles	CrossAttention	68.35	65.20	67.81	71.00	89.46	72.36
Cji	Principles	CrossAttention	**68.82**	65.20	67.98	71.40	88.01	72.28
BTS	Principles	CrossAttention	67.99	64.70	66.67	71.50	88.97	71.97
Character	Principles	CrossAttention	66.00	59.30	63.37	69.40	88.40	69.29
Subword	Principles	CrossAttention	59.19	54.50	67.62	69.70	88.57	67.92
Word	Principles	CrossAttention	66.48	61.10	62.82	72.00	88.71	70.22
Jamo	Linear	CrossAttention	67.04	57.70	64.35	69.30	88.11	69.30
Jamo	Attention	CrossAttention	68.14	64.70	67.28	71.60	88.32	72.01
Jamo	Principles	Summation	68.27	65.50	67.98	70.60	89.07	72.28
Jamo	Principles	Concatenation	66.55	61.10	66.61	69.90	89.06	70.64

Table 3: Ablation results for various architecture of **SCRIPT** applied to $\text{KoGPT2}_{\text{base}}$. The first row presents the best-performing variant. Cells corresponding to ablated components are highlighted in light blue. The global-best results are highlighted in **boldface**.

using excessively fine-grained units such as BTS leads to a slight performance drop, suggesting limitations in compressing overly fine-grained representations into higher-level coarse units. Furthermore, when we extended the comparison to larger units beyond the subcharacter level, including character, subword, and word units, performance degraded substantially. This finding suggests that our proposed method is specifically designed to preserve the compositional structure of subcharacters and is therefore less suited to larger linguistic units. Notably, using subword units, also employed in the base PLM, resulted in the largest performance drop. This result indicates that the observed gains are not simply attributable to increased parameter count or additional fusion capacity, but are instead driven by Jamo-level structural information.

Compression Method of Subcharacters in SCRIPT. Replacing the proposed composition-principled compression with generic pooling methods (Attention (Dai et al., 2020) or Linear (Nawrot et al., 2022)) leads to a 1.8–4.6%p performance drop, underscoring the importance of preserving the hierarchical compositional structure of *Hangul* during subcharacter aggregation.

Integration Method of Subword Representations. We further compare integration strategies between subcharacter and subword representations. CrossAttention (Vaswani et al., 2017) yields the strongest results, outperforming Summation and Concatenation, suggesting that dynamic alignment with PLM representations is key to integrating two heterogeneous subword representations effectively.

Overall, these results demonstrate that **SCRIPT**'s gains arise from explicitly encoding *Hangul*'s compositional structure and aligning it with pre-trained representations, rather than from any single architectural choice.

4.4 Effect of Tokenization Granularity on SCRIPT

Beyond conducting ablations on individual components of the **SCRIPT** architecture (§4.3), we also observed that the integration and effectiveness of compositional knowledge vary depending on the PLM's tokenization scheme. To investigate this, we compared four tokenization strategies: word, morpheme, subword, and character. As the final compositional token units changed accordingly, we also adjusted **SCRIPT**'s compressed output token unit to match. Thus, instead of the original "Character-to-Subword" setting described in Section 3.2.2, we experimented with "Character-to-Word," "Character-to-Morpheme," and "Character-to-Character."[5]

As shown in Table 4, **SCRIPT** proved effective when applied with larger units, such as Word and Morpheme, compared to Subword. This suggests that when the base PLM has already captured sufficient semantic meaning (Aguilar et al., 2021; Kaushal and Mahowald, 2022), integrating syntactic compositional knowledge leads to a synergistic improvement. In contrast, with the smaller Char-

[5]To minimize OOV occurrences, we constructed vocabularies based on prior work (Park et al., 2020; Kim et al., 2024), setting vocabulary sizes to 64k for Word, 32k for Morpheme and Subword, and 2k for Character.

Model	PLM Tokenization	KoBEST					Avg.
		BoolQ	COPA	WiC	HellaSwag	SentiNeg	
$BERT_{base}$	Word	60.04	57.60	62.70	55.00	52.39	57.55
$BERT_{base}$ + **SCRIPT**$_{Jamo}$		60.47	57.60	64.76	57.20	52.39	58.48(▲ 0.94)
$BERT_{base}$	Morpheme	63.75	58.50	71.75	61.80	78.59	66.88
$BERT_{base}$ + **SCRIPT**$_{Jamo}$		65.03	60.00	**72.06**	62.20	80.35	67.93(▲ 1.05)
$BERT_{base}$	Subword	67.22	67.10	68.90	69.10	88.50	72.16
$BERT_{base}$ + **SCRIPT**$_{Jamo}$		**68.28**	**68.20**	70.90	**72.40**	**89.47**	**73.85**(▲ **1.69**)
$BERT_{base}$	Character	62.89	61.00	71.35	48.60	78.84	64.54
$BERT_{base}$ + **SCRIPT**$_{Jamo}$		61.04	59.30	71.35	49.40	78.34	63.89(▼ 0.65)

Table 4: Comparison of the effectiveness of compositional knowledge integration into PLMs across different tokenization methods. The global-best results are highlighted in **boldface** and local-best results for each section are highlighted in underline, respectively.

acter unit, where the base PLM primarily learns syntactic rather than semantic knowledge (Aguilar et al., 2021; Mielke et al., 2021), applying **SCRIPT** introduced noise and hindered performance.

5 In-Depth Analysis

Beyond the quantitative results, this section offers a detailed linguistic analysis of how **SCRIPT** operates in Korean. We examine how **SCRIPT** enriches subword representations and enables the base model to more effectively capture key linguistic phenomena.

5.1 Impact on Morphological Variations

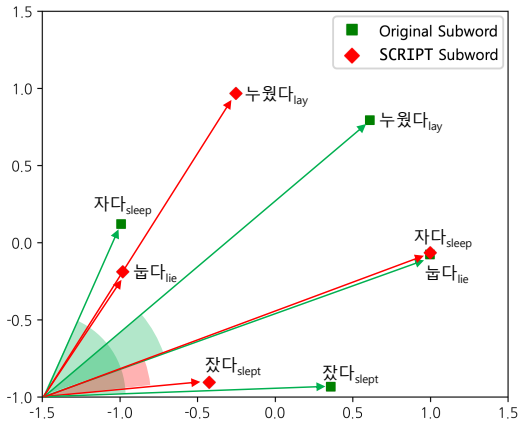


Figure 4: PCA visualization of subword embeddings for word pairs exhibiting subcharacter-level alternations. Each pair (e.g., 자다$_{sleep}$–잤다$_{slept}$, 눕다$_{lie}$–누웠다$_{lay}$) shares the same root meaning but differs in tense.

To assess how well **SCRIPT** captures subcharacter-level morphological alternations, we compare two subword representations: one from the PLM's original subword embeddings and the other from **SCRIPT**'s subcharacter-based representations. Using these, we represent morphologically related word pairs, such as tense-inflected forms. Figure 4 provides a qualitative geometric illustration using mean-centered embeddings projected via PCA. In the projected space, **SCRIPT** places morphologically related forms in closer angular proximity, indicating a more structured encoding of tense relationships, while the original PLM embeddings appear more dispersed. To verify that this pattern is not an artifact of 2D projection, we additionally compute cosine similarity in the original embedding space over 50 verb–past tense pairs, observing a consistent increase from 0.71 to 0.80 (+11%). These results suggest that subcharacter compositionality improves the model's ability to capture fine-grained grammatical variations in Korean.

5.2 Impact on Word Embedding Cohesion

As shown in Figure 5, we examine how Korean LMs organize semantically related predicate inflections in embedding space using five forms of the predicate '춥다$_{be\ cold}$'. Larger subword-based LMs show increasingly cohesive clustering, while the Jamo-based model, KOMBO (Kim et al., 2024), exhibits a more scattered distribution, likely due to its extreme focus on syntactic granularity. For clarity, we visualize KoGPT2 as a representative backbone, where **SCRIPT** produces the most compact grouping even at a small scale. Consistent patterns are observed across larger backbones. Together with ablation results showing degraded cohesion when compositional encoding is removed or altered, this suggests that the observed structure primarily arises from subcharacter compositional modeling rather than normalization effects.

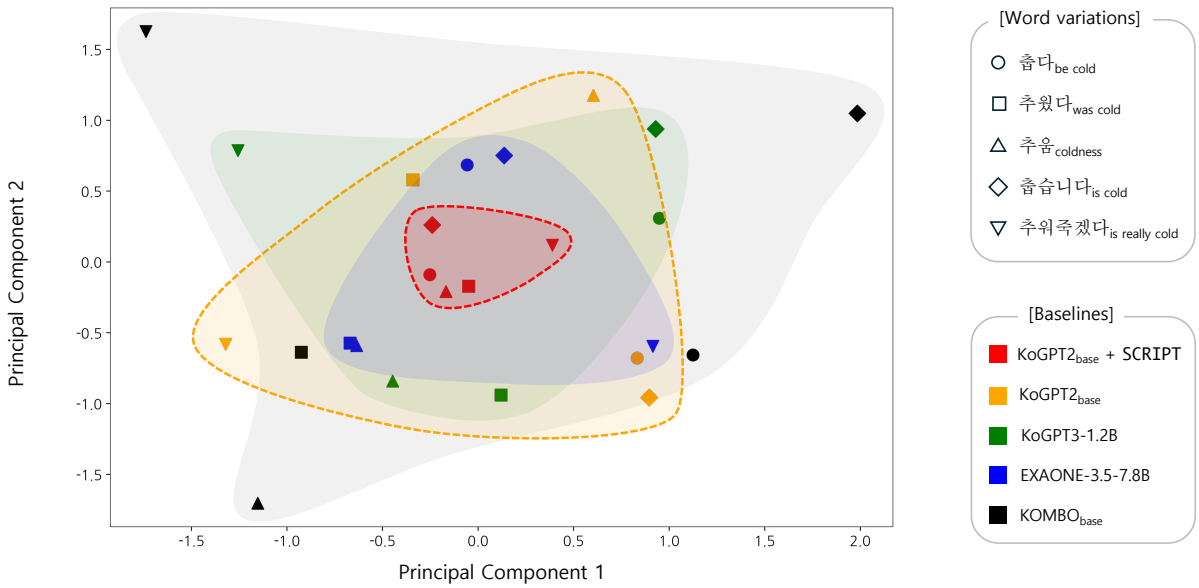


Figure 5: PCA visualization of word embeddings averaged over tokens for five semantically related Korean predicate inflections derived from the predicate '춥다$_{\text{be cold}}$'. The dashed boundaries indicate the dispersion ranges of the smallest baseline, KoGPT2$_{\text{base}}$, and its **SCRIPT**-augmented counterpart.

6 Related Work

6.1 Korean Pre-trained Language Models

Most off-the-shelf Korean PLMs employ subword-based tokenization (Yoo et al., 2024; LG AI Research et al., 2024), which has proven effective for handling Korean's rich morphology. However, such subword-based tokenizations do not explicitly model subcharacter-level structure, where many morphophonological processes in Korean occur. To address this limitation, several studies have explored Jamo-level modeling of Korean (Moon and Okazaki, 2020; Cognetta et al., 2023; Kim et al., 2024). In particular, Kim et al. (2024) explicitly encodes the compositional structure of *Hangul* to enrich character representations. However, these approaches typically rely on non-standard or encoder-only architectures and require full pre-training from scratch, which limits their applicability to general-purpose PLMs.

6.2 Multi-Granular Representations

Some prior work has attempted to incorporate multiple granularities in Korean language modeling, such as combining Jamo and word embeddings (Kwon et al., 2021) or switching between Jamo and subwords depending on context (Lee et al., 2025). However, these methods typically alternate between token levels rather than structurally integrating them. Moreover, they often lack architectural generality or remain limited to encoder-based tasks. Despite growing evidence that combining fine-grained morphological cues with higher-level representations improves performance (Lai et al., 2021; Zhao et al., 2023; Wang et al., 2024), Korean PLMs still underexplore this integration in a principled and efficient manner.

7 Conclusion

This work presents **SCRIPT**, a modular framework for injecting subcharacter compositional knowledge into Korean PLM. Through a structure-aware compression mechanism grounded in the compositional principles of *Hangul*, **SCRIPT** captures morphophonological variations at the subcharacter-level and enriches coarse PLM's token representations. Our experiments demonstrate that **SCRIPT** generally improves model performance across a wide range of Korean NLU and NLG tasks, enriching both conventional subword- and subcharacter-based approaches. Beyond quantitative gains, our linguistic analyses show that **SCRIPT** enhances the semantic and grammatical organization of the embedding space, enabling more cohesive clustering of inflected predicates and more faithful modeling of Korean linguistic phenomena. These findings underscore the limitations of subword tokenization in morphologically rich language and advocate for subcharacter-aware modeling as a necessary extension for Korean NLP.

Limitations

This work focuses on improving Korean language understanding by explicitly modeling structural characteristics specific to Korean. Accordingly, **SCRIPT** is primarily designed and evaluated within the Korean linguistic context, and its effectiveness for other languages is not systematically validated in this study. Although the modular design may be adaptable to languages with rich internal character structure or complex morphology, such extensions are beyond the scope of this paper. As a minimal proof of concept, we show that **SCRIPT** can be integrated into a multilingual pre-trained model and still improves Korean task performance (Appendix G). However, this experiment does not establish general cross-lingual applicability.

Second, while **SCRIPT** demonstrates consistent improvements across models ranging from approximately 100M to 2.4B parameters, we do not evaluate models at larger scales (e.g., 7B+). As larger models develop stronger internal representations, the relative benefit of explicitly modeling subcharacter structure may vary. Evaluating **SCRIPT** on larger-scale models remains an important direction for future work.

Finally, **SCRIPT** introduces additional parameters and sequence-length-dependent computations in the embedding layer, particularly when using cross-attention. Although this leads to improved convergence and performance, it also increases inference cost. Future work may explore more lightweight variants that better balance efficiency and effectiveness.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2025-00517221 and No.RS-2024-00415812) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2024-00439328, Karma: Towards Knowledge Augmentation for Complex Reasoning (SW Starlab), No.RS-2024-00457882, AI Research Hub Project, and No.RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University)).

References

Gustavo Aguilar, Bryan McCann, Tong Niu, Nazneen Rajani, Nitish Shirish Keskar, and Thamar Solorio. 2021. Char2Subword: Extending the subword embedding space using robust character compositionality. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1640–1651, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Marco Cognetta, Sangwhan Moon, Lawrence Wolfsonkin, and Naoaki Okazaki. 2023. Parameter-efficient Korean character-level language modeling. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2350–2356, Dubrovnik, Croatia. Association for Computational Linguistics.

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.

Florian Coulmas. 2003. *Writing Systems: An introduction to Their Linguistic Analysis*. Cambridge University Press.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. 2020. Funnel-transformer: filtering out sequential redundancy for efficient language processing. In *Proceedings of the 34th International Conference on*

Neural Information Processing Systems, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Peter Daniels and William Bright. 1996. *The World's Writing Systems*. Oxford University Press.

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 516 others. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Myeongjun Jang, Dohyung Kim, Deuk Sin Kwon, and Eric Davis. 2022. KoBEST: Korean balanced evaluation of significant tasks. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3697–3708, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Jongho Jun. 2018. Morpho-phonological processes in korean.

Ayush Kaushal and Kyle Mahowald. 2022. What do tokens know about their characters and how do they know it? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2487–2507, Seattle, United States. 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.

SungHo Kim, Nayeon Kim, Taehee Jeon, and SangKeun Lee. 2025. Polishing every facet of the GEM: Testing linguistic competence of LLMs and humans in Korean. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9955–9984, Vienna, Austria. Association for Computational Linguistics.

SungHo Kim, Juhyeong Park, Yeachan Kim, and SangKeun Lee. 2024. KOMBO: Korean character representations based on the combination rules of subcharacters. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5102–5119, Bangkok, Thailand. Association for Computational Linguistics.

Ohjoon Kwon, Dohyun Kim, Soo-Ryeon Lee, Junyoung Choi, and SangKeun Lee. 2021. 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*, pages 3213–3221, Online. Association for Computational Linguistics.

Yuxuan Lai, Yijia Liu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2021. Lattice-BERT: Leveraging multi-granularity representations in Chinese pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1716–1731, Online. Association for Computational Linguistics.

Iksop Lee and S Robert Ramsey. 2001. The korean language.

Junyoung Lee, Marco Cognetta, Sangwhan Moon, and Naoaki Okazaki. 2025. Jamo-level subword tokenization in low-resource Korean machine translation. In *Proceedings of the Eighth Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT 2025)*, pages 66–80, Albuquerque, New Mexico, U.S.A. Association for Computational Linguistics.

Kyong-Nim Lee and Minhwa Chung. 2003. Modeling cross-morpheme pronunciation variations for korean large vocabulary continuous speech recognition. In *INTERSPEECH*, pages 261–264.

LG AI Research, Soyoung An, Kyunghoon Bae, Eunbi Choi, Kibong Choi, Stanley Jungkyu Choi, Seokhee Hong, Junwon Hwang, Hyojin Jeon, Gerrard Jeongwon Jo, Hyunjik Jo, Jiyeon Jung, Yountae Jung, Hyosang Kim, Joonkee Kim, Seonghwan Kim, Soyeon Kim, Sunkyoung Kim, Yireun Kim, and 14

others. 2024. EXAONE 3.5: Series of large language models for real-world use cases. *Preprint*, arXiv:2412.04862.

Andrew Matteson, Chanhee Lee, Youngbum Kim, and Heuiseok Lim. 2018. Rich character-level information for Korean morphological analysis and part-of-speech tagging. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2482–2492, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, and 1 others. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*.

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.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

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. https://github.com/e9t/nsmc.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI spring symposium series*.

Geoffrey Sampson. 2015. *Writing Systems*. Equinox Publishing Limited.

Jaehyung Seo, Seounghoon Lee, Chanjun Park, Yoonna Jang, Hyeonseok Moon, Sugyeong Eo, Seonmin Koo, and Heuiseok Lim. 2022. A dog is passing over the jet? a text-generation dataset for Korean commonsense reasoning and evaluation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2233–2249, Seattle, United States. Association for Computational Linguistics.

Jiyoung Shin, Chi-yŏng Sin, Jieun Kiaer, Chae-ŭn Ch'a, and Jaeeun Cha. 2012. *The sounds of Korean*. Cambridge University Press.

Oleh Shliazhko, Alena Fenogenova, Maria Tikhonova, Anastasia Kozlova, Vladislav Mikhailov, and Tatiana Shavrina. 2024. mGPT: Few-shot learners go multilingual. *Transactions of the Association for Computational Linguistics*, 12:58–79.

Ho-Min Sohn. 2001. *The Korean language*. Cambridge University Press.

Unicode Consortium, editor. 2019. *The Unicode Standard, Version 12.0 – Core Specification*. Unicode, Inc., Mountain View, CA. Core specification PDF.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Yilin Wang, Xinyi Hu, and Matthew Gormley. 2024. Learning mutually informed representations for characters and subwords. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3201–3213, Mexico City, Mexico. 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.

Jaehoon Yeon and Lucien Brown. 2013. *Korean: A Comprehensive Grammar*, 2 edition. Routledge.

Kang Min Yoo, Jaegeun Han, Sookyo In, Heewon Jeon, Jisu Jeong, Jaewook Kang, Hyunwook Kim, Kyung-Min Kim, Munhyong Kim, Sungju Kim, Donghyun Kwak, Hanock Kwak, Se Jung Kwon, Bado Lee, Dongsoo Lee, Gichang Lee, Jooho Lee, Baeseong Park, Seongjin Shin, and 377 others. 2024. HyperCLOVA X technical report. *Preprint*, arXiv:2404.01954.

Soyoung Yoon, Sungjoon Park, Gyuwan Kim, Junhee Cho, Kihyo Park, Gyu Tae Kim, Minjoon Seo, and Alice Oh. 2023. Towards standardizing Korean grammatical error correction: Datasets and annotation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6713–6742, Toronto, Canada. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating text generation with BERT. *arXiv preprint arXiv:1904.09675*.

Shan Zhao, ChengYu Wang, Minghao Hu, Tianwei Yan, and Meng Wang. 2023. MCL: Multi-granularity contrastive learning framework for chinese ner. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 14011–14019.

A Composition of *Hangul* Characters

Hangul characters are constructed by combining initial consonants (Choseong), medial vowels (Jungseong), and, optionally, final consonants (Jongseong). The following outlines the possible components for each position:

Initial Consonants (Choseong)

The initial position can be occupied by one of the following 19 consonants:

ㄱ ㄲ ㄴ ㄷ ㄸ ㄹ ㅁ ㅂ ㅃ ㅅ ㅆ
ㅇ ㅈ ㅉ ㅊ ㅋ ㅌ ㅍ ㅎ

Medial Vowels (Jungseong)

The medial position consists of 21 vowels, which can be categorized based on their placement relative to the initial consonant:

- **Vertical vowels** (placed to the right of the initial consonant): ㅏ, ㅑ, ㅓ, ㅕ, ㅣ, ㅐ, ㅒ, ㅔ, ㅖ
- **Horizontal vowels** (placed below the initial consonant): ㅗ, ㅛ, ㅜ, ㅠ, ㅡ
- **Complex vowels** (combining both vertical and horizontal elements): ㅘ, ㅙ, ㅚ, ㅝ, ㅞ, ㅟ, ㅢ

These three categories correspond to the examples illustrated in Figure 6. This classification results in three distinct types of character shapes, each reflecting the spatial arrangement dictated by the vowel's orientation.

Final Consonants (Jongseong)

The final position may be unoccupied or contain one of the following 27 consonant combinations. _ means the empty final consonant and it is also considered a valid configuration:

ㄱ ㄲ ㄳ ㄴ ㄵ ㄶ ㄷ ㄹ ㄺ ㄻ
ㄼ ㄽ ㄾ ㄿ ㅀ ㅁ ㅂ ㅄ ㅅ ㅆ
ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ _

Syllable Structure

A typical *Hangul* syllable block is formed in one of the following structures:

- **CV**: Consonant + Vowel (e.g., 가)
- **CVC**: Consonant + Vowel + Consonant (e.g., 갈)

The positioning of vowels within the syllable block depends on their type:

- **Vertical vowels** are placed to the right of the initial consonant.
- **Horizontal vowels** are placed below the initial consonant.
- **Complex vowels** may occupy both right and bottom positions relative to the initial consonant.

This systematic arrangement allows for the construction of 11,172 possible syllable combinations in *Hangul*.

B Details for Inflection Frequency Evaluation

Action Definition. The alignment oracle (Matteson et al., 2018) aligns surface forms with lemma sequences by assigning one or more actions to each input character:

- **KEEP**: retain the character unchanged.
- **MOD**: modify the character into a different form (e.g., adding a final consonant, changing a vowel, or altering the initial consonant).
- **NOOP**: drop the character, i.e., it does not appear in the lemma.

Each action is augmented with BIO prefixes: "B-" marks the beginning of a morpheme, while "I-" denotes continuation within a morpheme. For instance, `B-KEEP` indicates the start of a morpheme where the character is preserved, while `B-MOD-`ㄴ signals a morpheme-internal modification introducing the consonant "ㄴ."

Corpus Preprocessing. We first parse an oracle-aligned action file in which each line contains a single input character and its action sequence (possibly multiple actions for one character), separated by a fixed delimiter. Non-*Hangul* characters are filtered out using a *Hangul* checker (we use `hgtk.checker.is_hangul` [6]). For each Korean character, we collect the raw action string and increment counters for `KEEP`, `MOD`, or `NOOP` depending on whether the action string contains these tokens. BIO prefixes (`B-`/`I-`) are preserved to indicate morpheme boundaries. (Matteson et al., 2018)

[6] https://pypi.org/project/hgtk/

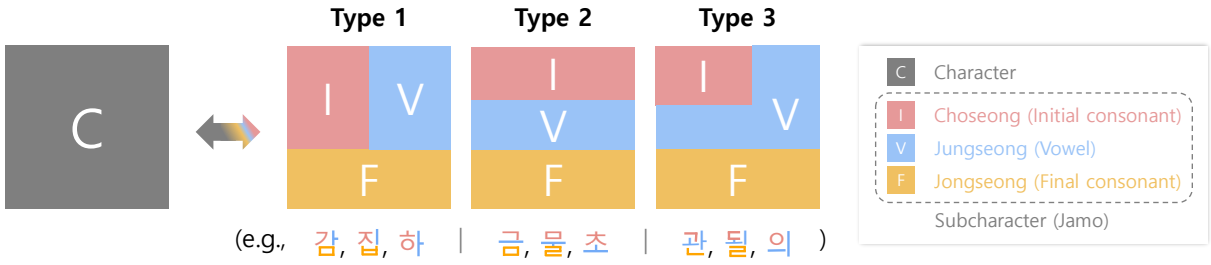


Figure 6: Three structural types of Korean syllable blocks, classified by the spatial arrangement of Choseong, Jungseong, and Jongseong.

Input Character	Oracle Actions	Output Lemma Units
런 (reon)	B-MOD-럽, I-MOD-ㄴ	럽 (reob), ㄴ (n)
했 (haess)	B-MOD-하, I-MOD-았	하 (ha), 았 (ass)
다 (da)	B-KEEP	다 (da)

Table 5: Examples of oracle actions aligned with lemma units. Each input character may correspond to multiple oracle actions (e.g., B-MOD, I-MOD) depending on the type and structure of the morphological transformation. The output shows the resulting lemma units produced by these actions.

Why Focus on MOD. Our analysis concentrated on "MOD" actions because they directly represent the sites of morphophonological change. "KEEP" characters reflect unchanged segments and "NOOP" characters denote deletions, both of which contribute little information about how Korean morphology operates. In contrast, "MOD" characters capture precisely the subcharacter or character-level transformations (e.g., tense, honorifics, adnominal endings) that are central to Korean grammar. By quantifying only "MOD," we obtain a clearer picture of where and how morphophonological alternations occur.

Granularity Classification (Subcharacter vs Character). We classify each `MOD` instance into two levels:

- **Subcharacter-level**: exactly one of {Choseong (I), Jungseong (V), Jongseong (F)} differs between the input and the aligned output, or a cross-syllable transfer/merge occurs, consistent with Korean fusion rules across character boundaries. (Matteson et al., 2018)
- **Character-level**: entire syllable box is replaced wholesale (non-comparable at the subcharacter-level).

In our implementation, we filter Korean characters (`hgtk.checker.is_hangul`) and compute counts of `KEEP`/`MOD`/`NOOP`. For `MOD` cases, we determine the granularity by comparing the input character's Unicode-decomposed (I, V, F) tuple with that of its aligned outputs. When a character yields multiple actions, we first reconstruct the immediate output units produced by that character's actions and then compare at the subcharacter-level. If only one subcharacter differs (or a Jongseong-Choseong transfer is observed), we mark it as subcharacter-level; otherwise, character-level.

NOOP Handling. `NOOP` marks deletions (input characters with no aligned lemma). As deletions indicate absence rather than transformation, we exclude `NOOP` from granularity statistics; counts are still reported for completeness. (Alternatively, `NOOP` can be treated as character-level; we opted for exclusion.)

C Implementation Details and Model Considerations

As shown in Figure 3(b), our proposed method, **`SCRIPT`**, can be simply plugged into the embedding layer, making it easy to apply to existing PLMs. This provides a model-agnostic advantage, allowing seamless integration with various architectures. As an extra implementation detail, unlike the decoder model, such as GPT, the encoder model, like BERT, has a particular design of the input sequence. BERT adds a special token `[CLS]` at the beginning of the input sequence, and this token is used as the representation of the sequence at the last-layer

Algorithm 1: Subword Representation with SCRIPT

Input: $\mathbf{s}$ (raw input sentence), embedding dimension D.
Output: $\mathbf{e}_{\mathrm{F}}$ (fused subword-level embeddings).

1 $\mathbf{t}_{\text{subchar}} \leftarrow \textsc{Tokenize}_{\text{subchar}}(\mathbf{s})$ // subcharacter sequence
2 $\mathbf{e} \leftarrow \textsc{Embedding}_{\text{subchar}}(\mathbf{t}_{\text{subchar}})$ // subcharacter representation
3 **if** first token in $\mathbf{e}$ is `[CLS]` **then** // (for encoder-only model)
$\mathbf{e}_{\text{CLS}} \leftarrow \mathbf{e}[1]$
$\mathbf{e} \leftarrow \mathbf{e}[2:N]$
4 $\mathbf{h} \leftarrow \text{GRU}(\mathbf{e})$ // sequential composition
5 **for** each for each integer k in [1, N/3]:
$\mathbf{h}_{\mathrm{I}} \leftarrow \mathbf{h}[3k-2]$ // Choseong
$\mathbf{h}_{\mathrm{V}} \leftarrow \mathbf{h}[3k-1]$ // Jungseong
$\mathbf{h}_{\mathrm{F}} \leftarrow \mathbf{h}[3k]$ // Jongseong
6 $\mathbf{h}_{\mathrm{I+V}} \leftarrow \text{GRU}(\mathbf{h}_I + \mathbf{h}_V)$ // fusion of Choseong and Jungseong
7 $\mathbf{h}_{\mathrm{R}} \leftarrow \textsc{Stack}(\mathbf{h}_I + \mathbf{h}_V, \mathbf{h}_F)$ // reshape subcharacter sequence
8 $\mathbf{h}_{\mathrm{C}} \leftarrow \textsc{AvgPool}(\textsc{Conv}_{2\times1}(\mathbf{h}_{\mathrm{R}}))$ // character representation
9 $\mathbf{h}_{\mathrm{S}} \leftarrow \textsc{Pooling}(\text{GRU}(\mathbf{h}_{\mathrm{C}}))$ // subword representation of **SCRIPT**

10 $\mathbf{t}_{\text{subword}} \leftarrow \textsc{Tokenize}_{\text{subword}}(\mathbf{s})$ // subword sequence
11 $\mathbf{e}_{\mathrm{S}} \leftarrow \textsc{Embedding}_{\text{subword}}(\mathbf{t}_{\text{subword}})$ // original subword representation

12 $\mathbf{e}_{\mathrm{F}} \leftarrow \textsc{CrossAttn}(\mathrm{Q}{=}\mathbf{e}_{\mathrm{S}}, \mathrm{K}{=}\mathbf{h}_{\mathrm{S}}, \mathrm{V}{=}\mathbf{h}_{\mathrm{S}})$ // fusion of subword representations
13 **if** $\mathbf{e}_{\text{CLS}}$ exists **then** $\mathbf{e}_{\mathrm{F}} \leftarrow [\mathbf{e}_{\text{CLS}}; \mathbf{e}_{\mathrm{F}}]$

return $\mathbf{e}_{\mathrm{F}}$

Table 6: The input text $\mathbf{s}$ is tokenized into subcharacter tokens (line 1) and subword tokens (line 10), respectively (§ 3.2). While structure-aware subword embeddings are built bottom-up from subcharacter to character to subword (lines 2-9), the original subword embeddings are looked up (lines 10-11). These two subword representation streams are fused by cross-attention to yield the final subword representation $\mathbf{e}_{\mathrm{F}}$ (lines 12-13) (§ 3.3).

hidden state. To preserve the special structure and meaning of the special token during the compression stage, we first separate the `[CLS]` token from the rest of the tokenized sequence (line 3 in Table 6). Then, we only use the remaining subcharacter sequence as input to **SCRIPT**. After processing through **SCRIPT**, we add the `[CLS]` hidden state back to the output of **SCRIPT**, the compressed subword representation (line 13 in Table 6). Formally, this can be represented as $\mathbf{h}_{\mathrm{S}} \in \mathbb{R}^{(N'+1)\times \mathrm{D}}$:

$$\mathbf{h}_{\mathrm{S}} = \textsc{Emb}_{\text{subchar}}(\mathbf{t}_0) \oplus \textbf{SCRIPT}(\mathbf{t}_{2:N+1}) \quad (5)$$

This trivial technique provides the versatility for our proposed methodology, **SCRIPT**, to be easily applied across all existing PLMs. In the following experimental section, we demonstrate the utility of our methodology by applying **SCRIPT** to both pre-trained encoder-only and decoder-only models.

Another important consideration is the integration of the two distinct subword representations, which come from different levels of granularity. This process is highly sensitive to normalization, as it can easily disrupt compatibility between the pre-trained subword embeddings and those derived from subcharacter representations. Notably, our empirical analysis shows that the pre-trained embeddings are approximately 30 times larger in size than those generated by **SCRIPT**. Normalizing these vectors to a common scale can distort the distribution of the pre-trained embeddings, with the randomly initialized subcharacter representations introducing significant noise and resulting in the loss of crucial learned information. Our analysis shows that structure-aware subword knowledge is gradually fine-tuned and transferred into the pre-trained embeddings during integration. Therefore, normalizing the two representations to the same scale

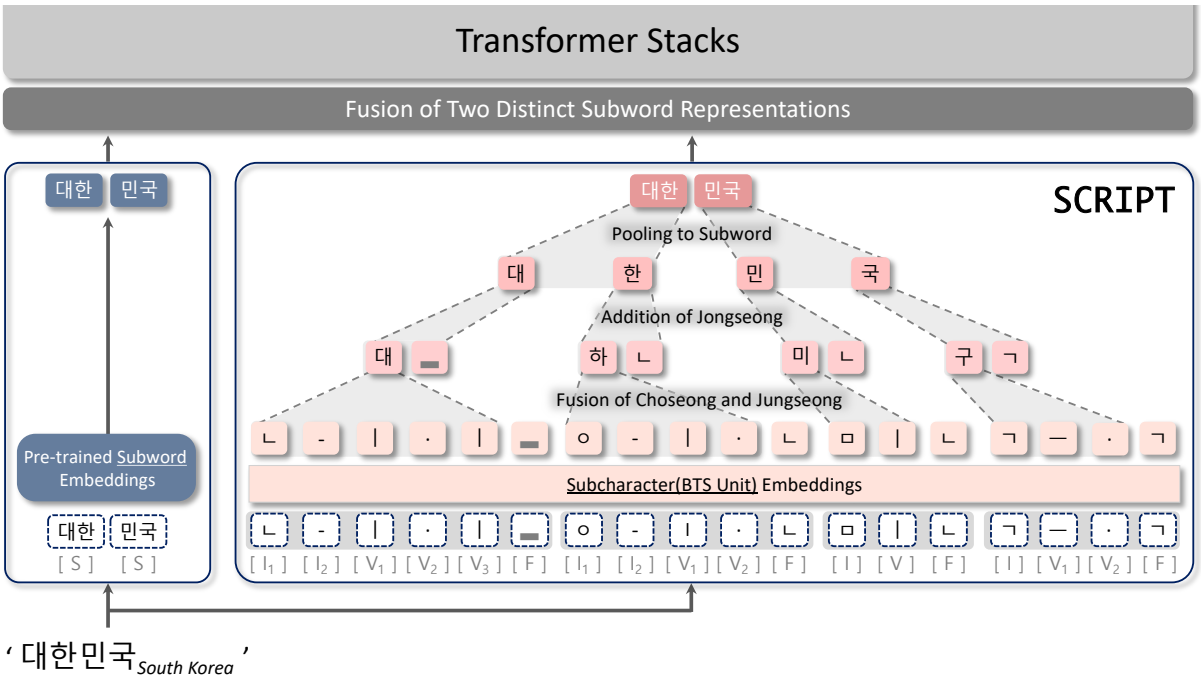


Figure 7: Implementation of **SCRIPT** for the BTS unit. This illustrates the hierarchical integration of subword representations derived from BTS unit in **SCRIPT**, using the example word '대한민국*South Korea*'. The word '대한민국*South Korea*' consists of two subwords ([S]: 대한, 민국), four characters (대, 한, 민, 국), and eighteen subcharacters: initial consonants ([I]: ㄴ, -, ㅇ, -, ㅁ, ㄱ), vowels ([V]: ㅣ, ·, ㅣ, ㅣ, ·, ㅣ, ㅡ, ·), and final consonants ([F]: ㅡ, ㄴ, ㄴ, ㄱ).

before fusion risks compromising the effectiveness of the pre-trained subword embeddings; it is empirically more effective to use them as they are.

D Implementation Details for BTS Units

Unlike Section 3.2, where we introduced the architecture of **SCRIPT** using Jamo as the base unit, this section extends the discussion to describe **SCRIPT** in terms of other subcharacters, such as BTS units. As noted in Section 3, BTS units are alternative subcharacter representations of *Hangul*, decomposing each character into even finer subcomponents than Jamo. According to Kim et al. (2022), consonants can be split into up to four subcomponents (e.g., the consonant 'ㅉ' decomposes into {ㅅ, -, ㅅ, -}, while vowels can be split into up to five subcomponents (e.g., the vowel 'ㅙ' decomposes into {·, ㅡ, ㅣ, ·, ㅣ}. Depending on the selective decomposition of consonants or vowels, there are three distinct types of units: consonant-only decomposition (denoted as Stroke), vowel-only decomposition (denoted as Cji, short for *Cheonjiin*), and both consonant and vowel decomposition (denoted as BTS). As a result, the maximum number of tokens per character varies depending on the decomposition type: the Stroke decomposition results in up to 9 subcharacter tokens, the Cji decomposition yields up to 7 tokens, and the BTS decomposition produces up to 13 tokens.

To incorporate the information of BTS units into the original subword representation of PLMs, we follow the progressive steps outlined in Section 3.2 in a similar manner. For simplicity, we employ BTS as the initial subcharacter unit of **SCRIPT** in the following explanation. We first tokenize the input text $\mathbf{s}$ into subcomponents, such as BTS, then project the resulting subcharacter sequence $\mathbf{t}_{\text{subchar}}$ into a subcharacter embedding space. A GRU layer is applied sequentially for contextualization:

$$\mathbf{t}_{\text{subchar}} = \textsc{Tokenize}_{\text{subchar}}(\mathbf{s}) \quad \in \mathbb{R}^{N} \tag{6}$$

$$\mathbf{e} = \textsc{Emb}_{\text{subchar}}(\mathbf{t}) \quad \in \mathbb{R}^{N \times D} \tag{7}$$

$$\mathbf{h} = \text{GRU}(\mathbf{e}) \quad \in \mathbb{R}^{N \times D} \tag{8}$$

Next, we merge the subcomponent tokens to construct representations for Choseong, Jungseong, and Jongseong. For each integer $k \in [1, N/13]$, the representations of Choseong ($h_{\text{I},k}$), Jungseong

($h_{V,k}$), and Jongseong ($h_{F,k}$) are defined as follows:

$$\mathbf{h}_{I,k} = \sum_{j=1}^{4} \mathbf{h}_{13(k-1)+j} \quad \in \mathbb{R}^{\frac{N}{13} \times D} \tag{9}$$

$$\mathbf{h}_{V,k} = \sum_{j=5}^{9} \mathbf{h}_{13(k-1)+j} \quad \in \mathbb{R}^{\frac{N}{13} \times D} \tag{10}$$

$$\mathbf{h}_{F,k} = \sum_{j=10}^{13} \mathbf{h}_{13(k-1)+j} \quad \in \mathbb{R}^{\frac{N}{13} \times D} \tag{11}$$

Next, we combine the representation of Choseong $\mathbf{h}_I$ and Jungseong $\mathbf{h}_V$:

$$\mathbf{h}_{I+V} = \mathbf{h}_I + \mathbf{h}_V \quad \in \mathbb{R}^{\frac{N}{13} \times D} \tag{12}$$

After that, we vertically concatenate the combined representation $\mathbf{h}_{I+V}$ with Jongseong representation $\mathbf{h}_F$:

$$\mathbf{h}_R = \begin{bmatrix} \mathbf{h}_{I+V} \\ \mathbf{h}_F \end{bmatrix} \quad \in \mathbb{R}^{2 \times \frac{N}{13} \times D} \tag{13}$$

To generate the dense character representation, we merge these vertically aligned representations by applying a convolution and a pooling layer:

$$\mathbf{h}_C = \text{AvgPool}(\text{Conv}(\mathbf{h}_R)) \quad \in \mathbb{R}^{\frac{N}{13} \times D} \tag{14}$$

Finally, a GRU layer is applied to the character representations $\mathbf{h}_C$, followed by a pooling layer to compress and align the granularity of the character representations with that of the original subword representations:

$$\mathbf{h}_S = \text{Pooling}(\text{GRU}(\mathbf{h}_C)) \quad \in \mathbb{R}^{N' \times D} \tag{15}$$

where N' represents the number of tokens of the original subword sequence. This compressed subword representation $\mathbf{h}_S$ is fused with the original subword representation $\mathbf{e}_S$ through cross-attention, as described in Section 3.3, to produce the final subword representation.

E Experimental Settings

E.1 Baselines

To evaluate the effectiveness of our proposed method, **SCRIPT**, we apply it to various PLMs listed below. Models equipped with our method are denoted as '`model`+**SCRIPT**'. When needed, the subcharacter type (e.g., Jamo) used in **SCRIPT** is indicated as a subscript, as in $\textbf{SCRIPT}_{\text{Jamo}}$. If no subscript is provided, Jamo is used by default.

- $\text{BERT}_{\text{base}}$ (Devlin et al., 2019): A bidirectional language model based on the Transformer architecture. It consists of multiple Transformer encoder layers. It is pre-trained in a self-supervised manner, enabling it to learn without labeled data. We utilize the $\text{BERT}_{\text{base}}$ model, which includes 12 Transformer encoder layers. It has a total of 110 million parameters. We employ a morpheme-aware tokenizer (Park et al., 2020) with a vocabulary size of 32k. We pre-train the $\text{BERT}_{\text{base}}$ model for 1 million steps on Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) tasks, using a corpus of 6.2GB consisting of the Korean Wikipedia and Namuwiki. [7]

- $\text{KOMBO}_{\text{base}}$ (Kim et al., 2024): A Jamo-based Korean encoder-only PLM that leverages the invention principles of *Hangul* to represent characters. While the architecture of $\text{KOMBO}_{\text{base}}$ is designed based on $\text{BERT}_{\text{base}}$, it differs in that it uses subcharacter-level tokens instead of subwords. It also includes a combination layer below and a restoration layer above its 12 Transformer blocks, introducing additional computational cost and overhead. However, it achieves better performance on NLU tasks than $\text{BERT}_{\text{base}}$. Moreover, since both $\text{BERT}_{\text{base}}$ and $\text{KOMBO}_{\text{base}}$ models are encoder-only models, they do not apply to generative tasks. We pre-train $\text{KOMBO}_{\text{base}}$ for 1 million steps on MLM and NSP tasks using a 6.2GB corpus from Korean Wikipedia and Namuwiki, following the same configuration as $\text{BERT}_{\text{base}}$.

- $\text{KoGPT2}_{\text{base}}$ [8]: A generative language model composed of multiple Transformer decoder blocks. Unlike the BERT model, which is trained on MLM and NSP tasks, GPT is trained on a next token prediction task, enabling it to generate contextually relevant text. KoGPT2 is a Korean variant of the GPT model, following the $\text{GPT2}_{\text{base}}$ configuration with 12 Transformer decoder blocks and 125 million parameters. It has been pre-trained on the Korean Wiki and Korpora datasets, totally over 40GB. KoGPT2 employs subword tokenization with a vocabulary size of 51.2k.

[7] https://namu.wiki/

[8] https://github.com/SKT-AI/KoGPT2

- KoGPT3-1.2B [9]: A large-scale Transformer decoder model containing 1.2 billion parameters and 24 Transformer decoder blocks. This model follows the GPT-3 architecture. The model is trained on Ko-DAT, a large-scale, curated Korean dataset created by SK Telecom with 35 billion tokens, using the next token prediction task over 72k training steps. Similar to $\text{KoGPT2}_{\text{base}}$, KoGPT3-1.2B uses subword tokenization with a 51.2k vocabulary size.

- mGPT-1.3B (Shliazhko et al., 2024): The multilingual extension of GPT-3 which is pre-trained across 61 languages. It consists of 24 Transformer decoder layers with a total of 1.3 billion parameters. Pre-training was conducted on the Wikipedia and C4 corpora, over 600GB in total, for 600k steps. The model employs a 100k size vocabulary and utilizes Byte-level Byte Pair Encoding (BBPE) as its default tokenization strategy, enhancing its multilingual capabilities.

- EXAONE-3.5-2.4B-Instruct (EXAONE-2.4B for short) (LG AI Research et al., 2024): A bilingual (Korean and English) instruction-tuned language model developed by LG AI Research. It uses a decoder-only Transformer architecture and is part of the EXAONE 3.5 series. The model has 30 Transformer decoder layers and 2.41 billion parameters total. It was trained under a causal/next-token prediction objective using a bilingual corpus curated by LG AI Research, supports a maximum context length of 32,768 tokens, and employs a shared vocabulary of 102,400 tokens using a BBPE tokenizer.

E.2 Implementation Details

We utilized a series of pre-trained GPT models, including $\text{KoGPT2}_{\text{base}}$, KoGPT3-1.2B, mGPT-1.3B, and EXAONE-2.4B all sourced from the Huggingface library[10], and fine-tuned them using LoRA (Low-Rank Adaptation) (Hu et al., 2021). However, for BERT-based models, such as $\text{BERT}_{\text{base}}$ and $\text{KOMBO}_{\text{base}}$, training with LoRA showed instability, so we opted for full fine-tuning exclusively for these models. As noted by Hu et al. (2021), there was no significant difference in performance between LoRA and full fine-tuning. Since we utilize the DeepSpeed library[11] for models larger than 1B parameters, all models were trained on a single NVIDIA RTX 3090 GPU. We set the default maximum sequence length to 256 for the original PLM's subword tokenizer, and to 2048 for the subcharacter tokenizer used in $\text{KOMBO}_{\text{base}}$ and $\textbf{SCRIPT}_{\text{Jamo}}$. For tasks requiring longer inputs, such as HellaSwag and XL-Sum, we used sequence lengths of 512 and 3072, respectively. We use the `AdamW` optimizer and cosine learning rate scheduler. For most other experimental settings, we used the default configurations of each pre-trained model. The detailed hyperparameter settings are summarized in Table 7. All experiments are repeated over 3 random seeds (42–44), and we report the mean.

E.3 Tasks

Korean NLU Tasks. To investigate the performance of our proposed method on Korean NLU tasks, we evaluated baselines on nine distinct Korean NLU datasets. Four of these, KorNLI, KorSTS, NSMC, and PAWS-X, are widely used benchmarks for Korean NLU tasks, which we refer to as "*Korean Standard NLU Tasks*" (Jang et al., 2022). The remaining five datasets belong to the KoBEST benchmark (abbreviated as KB), which is designed to evaluate Korean language models on more complex linguistic understanding. We refer to these as "*Korean Advanced NLU Tasks*" (Jang et al., 2022).

- KorNLI (Ham et al., 2020): A dataset comprising 943k train, 25.5k validation, and 5k test samples for NLI, derived from the SNLI (Bowman et al., 2015), MNLI (Williams et al., 2018), and XNLI (Conneau et al., 2018) datasets. The data is labeled across three classes: entailment, neutral, and contradiction.

- KorSTS (Ham et al., 2020): A dataset developed to assess the semantic similarity between sentence pairs, adapted from the Korean STS-B dataset (Cer et al., 2017). KorSTS consists of 5,749 train samples and 2,879 evaluation samples, each labeled with a similarity score from 0 to 5, indicating the degree of semantic similarity between the sentences.

- NSMC (Park, 2016): A dataset sourced from NAVER is used for sentiment analysis of Ko-

[9] https://huggingface.co/skt/ko-gpt-trinity-1.2B-v0.5

[10] https://huggingface.co/models

[11] https://github.com/deepspeedai/DeepSpeed

Task	Epoch	Batch Size	Learning Rate	Dropout Ratio	Warmup Ratio	LoRA r	LoRA α
KorNLI	5	64	BERT: {5e-05, 1e-04}	0.03	0.1	32	128
KorSTS	15	64	GPT : {1e-05, 5e-05, 1e-04,				
NSMC	5	64	1e-03, 3e-03, 1e-02}				
PAWS-X	10	64					
BoolQ	10	8	BERT: {1e-05, 5e-05}	0.03	0.1	32	128
COPA	15	16	GPT : {1e-05, 5e-05, 1e-04,				
WiC	15	16	1e-03, 3e-03, 1e-02}				
HellaSwag	10	8					
SentiNeg	10	64					
KoCommonGen	15	64	GPT : {1e-04, 1e-03, 1e-02,	0.03	0.1	32	128
XL-Sum	10	64	2e-02, 3e-02, 4e-02}				
Kor-Learner	10	64					
Kor-Native	10	64					

Table 7: Hyperparameters used in all experiments in this paper for each task. For the learning rate, we select the value that yields the best performance for each baseline. Here, "BERT" refers to encoder-only models, including BERT and KOMBO, while "GPT" encompasses all decoder-only models, such as KoGPT2, KoGPT3, mGPT, and EXAONE.

rean movie reviews. It includes 150k train samples and 50k test samples, with each review labeled as either negative or positive.

- PAWS-X (Yang et al., 2019): A dataset for paraphrase identification, which includes six different language tasks. We only use the Korean subset to evaluate models. It contains 53k sentence pairs (49k for train, 2k for development, and 2k for test), each data labeled with one of two values: different meanings or paraphrases.

- KoBEST (Jang et al., 2022): A benchmark suite designed to evaluate broad linguistic and cognitive capabilities of Korean language models through five diverse and challenging tasks.
  - KB-BoolQ (Jang et al., 2022): A dataset of 3.7k train, 700 validation, and 1.4k test instances. The task is a true/false question and answer format based on paragraphs, with sources from Korean Wikipedia.
  - KB-COPA (Jang et al., 2022): A dataset includes 3.1k train, 1k validation, and 1k test instances. Models predict cause or effect given a premise, designed similarly to the English COPA dataset (Roemmele et al., 2011).
  - KB-WiC (Jang et al., 2022): A dataset contains 3.3k train, 1.3k validation, and 1.3k test samples, requiring models to determine if a target word holds the same meaning across two contexts.
  - KB-HellaSwag (Jang et al., 2022): A dataset composed of 2k train, 500 validation, and 500 test examples, where models select the most probable sentence to follow a given context. The data is sourced from YouTube and Wikipedia.
  - KB-SentiNeg (Jang et al., 2022): A dataset for sentiment analysis (3.6k train, 400 validation, 397 test samples) focusing on the polarity of negated sentences in product reviews.

Korean NLG Tasks. We used three distinct benchmarks, KoCommonGen, XL-Sum, and Korean Grammatical Error Correction (GEC), to evaluate the performance of our proposed method on Korean NLG tasks. The detailed explanations of each benchmark are provided below:

- KoCommonGen (Seo et al., 2022): A generative commonsense reasoning dataset comprising 43,188 train samples and 2,040 test examples. Given a set of morphemes, the model composes a sentence that reflects commonsense knowledge.

- XL-Sum (Hasan et al., 2021): A summarization dataset used to evaluate models' ability to generate concise and accurate summaries from large text bodies. We focus on the Korean subset, which includes 4,407 train samples and 550 validation and test samples. We are only using

Model	XL-Sum						
	BLEU 3	BLEU 4	ROUGE-2	ROUGE-L	METEOR	mBERTScore	KoBERTScore
$\text{KoGPT2}_{\text{base}}$	7.27	4.98	12.91	26.83	13.43	76.22	88.97
$\text{KoGPT2}_{\text{base}}$ + **SCRIPT**	7.64	5.20	13.30	27.24	13.67	76.31	89.20
KoGPT3-1.2B	9.14	6.21	15.88	30.18	16.91	76.95	89.69
KoGPT3-1.2B + **SCRIPT**	**9.39**	**6.40**	**15.99**	**30.31**	**16.97**	**77.46**	**89.77**

Table 8: Performance on XL-Sum multilingual summarization task. We evaluate only on the Korean summarization dataset. We use seven automatic evaluation metrics, including n-gram-based measures like BLEU, ROUGE, and METEOR; and two BERT-based scores (Zhang et al., 2019), mBERTScore and KoBERTScore, for semantic similarity. The global-best results are highlighted in **boldface** and local-best results for each model are highlighted in underline, respectively.

Model	Kor-Learner				Kor-Native			
	M^2_{pre}	M^2_{rec}	$M^2_{F_{0.5}}$	GLEU	M^2_{pre}	M^2_{rec}	$M^2_{F_{0.5}}$	GLEU
$\text{KoGPT2}_{\text{base}}$	29.35	16.11	25.19	21.60	72.12	55.19	67.76	61.45
$\text{KoGPT2}_{\text{base}}$ + **SCRIPT**	30.02	16.83	25.34	23.54	72.95	56.15	69.00	62.25
KoGPT3-1.2B	47.05	23.01	38.89	35.22	84.76	69.54	81.20	75.21
KoGPT3-1.2B + **SCRIPT**	**49.45**	**26.36**	**41.68**	**39.45**	**85.49**	**70.02**	**81.87**	**75.40**

Table 9: Performance on two Korean GEC tasks. As the evaluation metrics, we use M^2 scorer (Dahlmeier and Ng, 2012), which measures precision, recall, and $F_{0.5}$ scores based on edits and GLEU (Napoles et al., 2015) for the simple n-gram matching. The global-best results are highlighted in **boldface** and local-best results for each model are highlighted in underline, respectively.

the Korean subset of XL-Sum dataset for our experiments.

- Korean GEC (Yoon et al., 2023): A grammatical error correction dataset for Korean. It consists of four sub-datasets: three standalone datasets, such as Kor-Learner, Kor-Lang8, and Kor-Native, and one aggregated dataset, Kor-Union. Kor-Learner offers a more structured and reliable dataset for the Korean GEC task, as it is annotated by Korean language tutors. In contrast, Kor-Lang8 was corrected by native speakers through an open online platform. In this paper, we focus on two orthogonal GEC tasks, such as Kor-Learner and Kor-Native, to more accurately evaluate model performance relative to dataset complexity.
  - Kor-Learner GEC (Yoon et al., 2023): A GEC dataset for Korean learner texts, containing 19,898 train sentences, 4,264 validation sentences, and 4,265 test sentences. Kor-Learner contains learner-written essays that have been carefully corrected and annotated by Korean tutors. It aids in identifying and correcting grammar errors specific to Korean language learners.
  - Kor-Native GEC (Yoon et al., 2023): A GEC dataset targeting native Korean texts to support advanced linguistic understanding. It comprises 12,292 train sentences, 2,634 validation sentences, and 2,634 test sentences.

F Evaluation on Generative Tasks

F.1 XL-Sum

As shown in Table 8, **SCRIPT** consistently demonstrated its effectiveness on n-gram metrics. However, the performance improvement observed in the summarization task was somewhat smaller compared to that in the commonsense generation task. This difference arises because the summarization task typically involves input and output sentences that are approximately five times longer than those in the commonsense generation task, such as Ko-CommonGen. This suggests that **SCRIPT** is particularly effective at generating concise, well-formed sentences, demonstrating its strength in handling shorter and more focused outputs.

F.2 Korean GEC

Our proposed method, **SCRIPT**, also demonstrated the strongest performance on Korean grammatical

Model	KoBEST					Avg.
	BoolQ	COPA	WiC	HellaSwag	SentiNeg	
KoGPT3-1.2B	77.32	**82.80**	72.78	78.90	96.31	81.62
KoGPT3-1.2B + **SCRIPT**	**77.63**	**82.80**	**74.65**	**79.30**	**96.48**	**82.17**
mGPT-1.3B	71.19	69.70	68.38	76.10	89.40	74.95
mGPT-1.3B + **SCRIPT**	70.72	70.60	69.17	76.70	90.42	75.52

Table 10: Performance of KoGPT3-1.2B and mGPT-1.3B on KoBEST benchmark. The evaluation metrics for each task are accuracy (%). The global-best results are highlighted in **boldface** and local-best results for each model are highlighted in underline, respectively.

Model	KoCommonGen							Avg.
	BLEU 3	BLEU 4	ROUGE-2	ROUGE-L	METEOR	mBERTScore	KoBERTScore	
KoGPT3-1.2B	26.19	17.20	58.85	62.53	52.11	85.41	91.17	56.21
KoGPT3-1.2B + **SCRIPT**	**28.89**	**19.58**	**59.28**	**64.80**	**52.37**	**86.26**	**91.78**	**57.57**
mGPT-1.3B	15.16	8.07	37.77	50.99	33.31	80.17	89.34	44.97
mGPT-1.3B + **SCRIPT**	16.59	9.11	39.31	52.68	34.77	80.82	89.95	46.18

Table 11: Performance of KoGPT3-1.2B and mGPT-1.3B on KoCommonGen dataset. We use eight automatic evaluation metrics: BLEU, ROUGE, and METEOR for n-gram-based measures; and mBERTScore and KoBERTScore for semantic similarity. The global-best results are highlighted in **boldface** and local-best results for each model are highlighted in underline, respectively.

error correction tasks. As shown in Table 9, it consistently outperformed the base model in both the Kor-Learner and Kor-Native tasks, showing particularly strong effectiveness in the Kor-Learner task with average improvements exceeding an average of 3.2%p gains over the global-best performing baseline. According to Yoon et al. (2023), the Kor-Learner task contains a large proportion of errors related to particles, endings, and conjugations compared to the Kor-Native task. As mentioned in Section 2, linguistic variations in Korean frequently occur at the subcharacter-level. Therefore, the substantial performance gains observed on the Kor-Learner task demonstrate the effectiveness of our core approach: integrating subcharacter compositional information into subword representations. This result further validates that **SCRIPT** is highly suitable for the Korean language and effectively enriches the PLM's subword representations.

G Effectiveness of Multilingual Model

Our proposed method, **SCRIPT**, can be seamlessly integrated into the embedding layer of any model and is broadly applicable in multilingual settings. To demonstrate its effectiveness for Korean in multilingual models, we evaluated it on both a Korean monolingual model and a multilingual model that supports Korean. Specifically, we used KoGPT3-1.2B as the monolingual baseline and mGPT-1.3B, a multilingual model with comparable architecture and scale. For the NLU task, we employed the KoBEST benchmark to assess knowledge understanding, and for the NLG task, we used KoCommonGen to evaluate complex knowledge generation, including commonsense reasoning.

As shown in Table 10 and Table 11, multilingual models with **SCRIPT** largely outperformed their base counterparts in both KoBEST (NLU) and KoCommonGen (NLG) tasks. Notably, mGPT-1.3B achieved a performance gain of approximately 0.6%p on KoBEST benchmarks and 1.2%p on KoCommonGen, closely mirroring the improvements observed in the monolingual model. Gains in generative tasks were nearly twice as large as those in understanding tasks, indicating that **SCRIPT** is particularly effective in enhancing generative capabilities for Korean. These findings highlight the promise of scaling up to significantly larger and more extensively pre-trained multilingual generative models, such as the Llama (Dubey et al., 2024) and Qwen (Qwen et al., 2025) series. In particular, this substantial improvement in multilingual mod-

els is especially valuable given the current scarcity of specialized pre-trained LLMs for Korean.

H Qualitative Analysis for Generations

To analyze the quality of machine-generated text, in Table 12, we conducted a qualitative analysis for each generative task by model. We established two baselines for comparison: the basic KoGPT2$_{base}$ model and KoGPT2$_{base}$+**SCRIPT**. This analysis covered all Korean NLG tasks performed in Section 4.2. Since the input text for the summarization task, XL-Sum, is quite long, we have included examples in the H.2 for further details if needed.

H.1 KoCommonGen

Given the set of morphemes as input to the model, it generates the sentence as output by including the morphemes. As a result of the experiments, shown in Table 12, we observed that KoGPT2$_{base}$+**SCRIPT** model correctly generated appropriate particles by identifying the characteristics and position of objects, such as rail, train, and road. In contrast, the baseline model, KoGPT2$_{base}$, incorrectly generated the position of the word 'train' as 'beside the tracks' instead of 'on the tracks'. In English, the difference between 'beside' and 'on' involves several letters, whereas in Korean, this distinction is very subtle, differing by only a single subcharacter, '니' ('옆의') for 'on' and ' 네' ('옆에') for 'beside', which makes it more challenging to distinguish. This shows that **SCRIPT** effectively captures subtle nuances at the subcharacter-level.

H.2 XL-Sum

The summarization performance appears similar, but the base model tends to generate slightly longer sentences. Overall, our proposed method produces more concise summaries. For example, in this task's sample data, while KoGPT2$_{base}$ focused on the 'act of collecting samples', including the 'lunar landing', our model emphasized the 'successful completion of the exploration', generating sentences with a clearer focus on summarization itself.

H.3 Kor-Learner

As mentioned earlier in Section 4.2, the Kor-Learner dataset contains a higher frequency of errors related to particles, endings, and conjugations compared to other Korean GEC datasets. The sample in Table 12 also requires corrections for grammatical errors in endings. While the KoGPT2$_{base}$ model failed to correct these properly, our proposed method successfully adjusted endings by considering their agreement with predicates. As shown in Figure 1, understanding these types of grammatical errors is particularly important for Korean. Thus, our proposed method is both suitable and essential for effectively handling Korean.

H.4 Kor-Native

Through examples from the Kor-Native task, we confirmed that our proposed method, **SCRIPT**, enhances the ability to handle whitespace and noun recognition effectively. As shown in the sample in Table 12, it asked to identify and correct the incorrect word '테니그$_{tennig}$' to the appropriate noun '테니스$_{tennis}$'. While the naive KoGPT2$_{base}$ model failed to detect the error in this sentence and thus cannot make the necessary correction, the model using our proposed **SCRIPT** method successfully identified and corrected wrong word to '테니스$_{tennis}$'. Although this adjustment involved only a subtle subcharacter-level difference, changing 'ㄱ$_{g}$' to 'ㅅ$_{s}$', it once again demonstrated that subword models using larger token units than character-level cannot adequately handle such distinctions.

I Impact of Fused Representations

To examine how compositional knowledge of subcharacters affects subword representations, we compare (i) subcharacter embeddings from **SCRIPT**, (ii) original subword embeddings from the PLM, and (iii) fused subword embeddings augmented by **SCRIPT**. As shown in Figure 8, **SCRIPT**'s subcharacter representations yield the highest similarity among the predicate inflected word sets sharing root semantics but differing at the subcharacter-level. Notably, this advantage transfers to the fused embeddings, which accurately capture these fine-grained relational patterns. These results underscore the effectiveness of the proposed module in modeling subcharacter-level variation through compositional and representational fusion.

Task	Lang	Example
KoCommonGen	Ko	**Input:** { 있, 선로, 길, 옆, 열차 } **Gold Label:** 길 **옆의** 선로**에** 열차가 있다. **KoGPT2:** 선로가 길 **옆의** 길 **옆에** 열차가 세워져 있다. **KoGPT2 + SCRIPT:** 열차들이 길 **옆의** 선로**에** 있다.
	En	**Input:** { be, tracks, road, beside, train } **Gold Label:** There is a train **on** the tracks **beside** the road. **KoGPT2:** The train is parked **beside** the track **beside** the road. **KoGPT2 + SCRIPT:** The trains are **on** the tracks **beside** the road.
Kor-Learner	Ko	**Input:** 그리고 가장 중요한 영향은 그 앞으로 그 여행으로 이전보다 훨씬 더 '처음' 을 접할 **거다**. **Gold Label:** 그리고 가장 중요한 영향은 앞으로 여행으로 이전보다 훨씬 더 '처음' 을 접할 **것이라는 사실이다**. **KoGPT2:** 그리고 가장 중요한 영향을 그 앞으로 그 여행으로 이전보다 훨씬 더 '처음' 을 접할 **거다**. **KoGPT2 + SCRIPT:** 그리고 가장 중요한 영향은 그 앞으로 그 여행으로 이전보다 훨씬 더 '처음' 을 접할 **거라는 것이다**.
	En	**Input:** And the most important impact **would** that, through that journey, they will encounter the 'first' much more than before. **Gold Label:** And the most important impact **is the fact** that, through future travels, they will encounter the 'first' much more than before. **KoGPT2:** And the most important impact **would** that, through that journey, they will encounter the 'first' much more than before. **KoGPT2 + SCRIPT:** And the most important impact **is the thing** that, through that journey, they will encounter the 'first' much more than before.
Kor-Native	Ko	**Input:** 주말에 함께 ***테니그**를 쳐여. **Gold Label:** 주말에 함께 **테니스**를 쳐요. **KoGPT2:** 주말에 함께 ***테니그**를 쳐요. **KoGPT2 + SCRIPT:** 주말에 함께 **테니스**를 쳐요.
	En	**Input:** Let's play ***tennig** together on the weekend. **Gold Label:** Let's play **tennis** together on the weekend. **KoGPT2:** Let's play ***tennig** together on the weekend. **KoGPT2 + SCRIPT:** Let's play **tennis** together on the weekend.

Table 12: Examples for the qualitative analysis of four generation tasks. For each task, examples are composed of the input provided to the model, the gold label, and the predictions generated by two baselines: $\text{KoGPT2}_{\text{base}}$ and $\text{KoGPT2}_{\text{base}}$ applied with **SCRIPT**. The model outputs were generated in Korean, with English translations provided alongside for clarity. The asterisk (*) indicates a ungrammatical word. Red-colored characters represent incorrect parts, while blue-colored characters indicate correct representations.

Task	Language	Example
XL-Sum	Korean	**Input:** 최종 목적은 2㎏ 정도의 '토양' 표본을 상승선, 귀환선에 전달해 지구까지 가져오는 것이다 중국국가우주국(CNSA)은 달의 암석과 토양 표본을 수집해 지구로 가져오기 위해 출발한 무인 달 탐사선 '창어 5호'가 1일 밤 착륙에 성공했다고 2일 밝혔다. 창어 5호는 '폭풍의 바다'(Oceanus Procellarum)라는 지역 내 '몽스 륌케르'(Mons Rümker) 화산지대 북쪽에 안착했다. 이곳에서 며칠간 달 표면의 흙과 암석 표본 등을 수집한다. 창어 5호 탐사선에는 작업을 돕기 위한 카메라, 레이더, 드릴, 삽 등이 탑재돼있다. 최종 목적은 2㎏ 정도의 표토 표본을 상승선과 궤도선을 거쳐 귀환선에 전달해 지구까지 가져오는 것이다. 달의 토양 표본을 지구로 가져온 탐사선은 44년 전 1976년 옛 소련의 루나 24호가 마지막으로, 당시 200g의 토양을 지구로 옮기는 데 성공했다. 창어 5호 프로젝트팀이 환호하는 모습 이날 달 착륙 모습은 일주일 전 발사 때와 달리 생중계 되지 않았다. 중국 TV 채널에서는 성공적인 착륙이 확인되고 나서야 정규 방송을 중단하고 이를 녹화 중계했다. 공개된 착륙 과정에는 탐사선의 다리가 달의 먼지 쌓인 표면에 그림자를 드리우는 장면 등이 포함됐다. ... **KoGPT2:** 중국 국가우주국이 달 착륙에 성공한 창어 6호 달 착륙선에 탑재된 카메라와 레이더를 통해 달 표면 표본을 지구까지 운반했다. **KoGPT2 + SCRIPT:** 중국의 달 탐사 프로젝트가 성공적으로 마무리됐다.
	English	**Input:** The primary goal is to bring approximately 2 kg of lunar soil samples back to Earth by transferring them from the ascent and return modules. The China National Space Administration (CNSA) announced on the 2nd that its unmanned lunar probe, Chang'e-5, successfully landed on the night of the 1st to collect lunar rock and soil samples to return to Earth. Chang'e-5 has landed in the volcanic area north of Mons Rümker within the region known as Oceanus Procellarum. Over the next few days, it will collect samples of lunar soil and rock. Equipped with cameras, radar, drills, and shovels to aid in its operations, the ultimate goal of Chang'e-5 is to gather about 2 kg of surface samples, which will be transferred from the ascent and orbital modules to the return module for their journey back to Earth. The last mission to bring lunar soil samples to Earth was the Soviet Union's Luna 24 in 1976, which successfully transported 200 g of lunar soil back to Earth. On the day of the landing, the Chang'e-5 team celebrated. Unlike the launch, the landing was not broadcast live, and Chinese TV channels interrupted regular programming to air recorded footage after confirming a successful landing. The released landing process included images of the lander casting a shadow on the dusty lunar surface. ... **Gold Label:** China has landed another probe on the surface of the moon. **KoGPT2:** The China National Space Administration successfully transported lunar surface samples to Earth using the camera and radar aboard the Chang'e 6 lunar lander. **KoGPT2 + SCRIPT:** China's lunar exploration project has been successfully completed.

Table 13: Examples for the qualitative analysis of XL-Sum task. For each task, examples are composed of the input provided to the model, the gold label, and the predictions generated by two baselines: $\text{KoGPT2}_{\text{base}}$ and $\text{KoGPT2}_{\text{base}}$ applied with **SCRIPT**. The model outputs were generated in Korean, with English translations provided alongside for clarity.

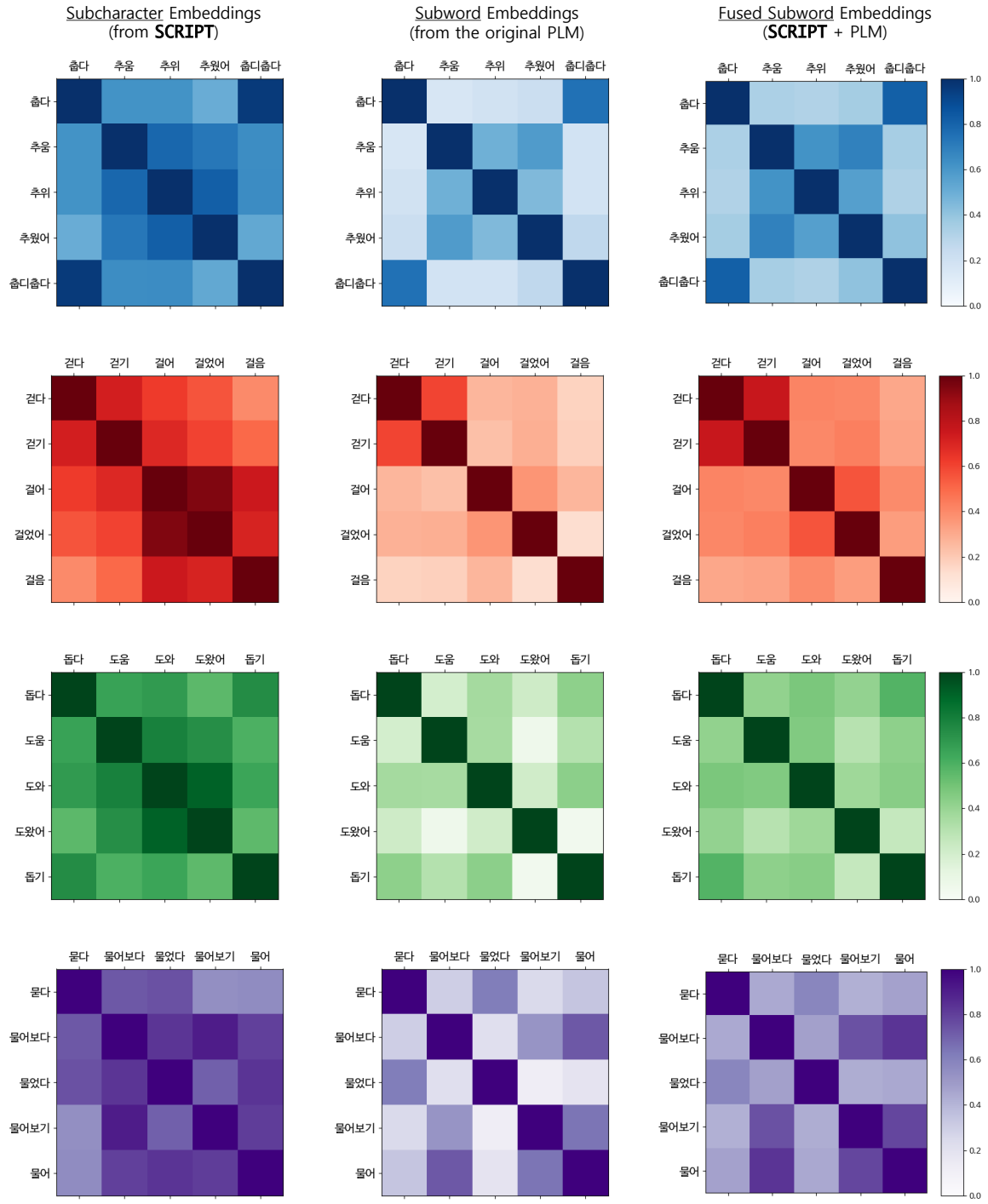


Figure 8: Visualization of the similarities between word representations, measured from conjugated word pairs. Each word set contains five words that share the same root meaning. The first word set, {'춥다', '추움', '추위', '추웠어', '춥디춥다'}, conveys the meaning 'cold'; the second set, {'걷다', '걷기', '걸어', '걸었어', '걸음'}, represents 'walk'; the third set, {'돕다', '도움', '도와', '도왔어', '돕기'}, signifies 'help'; and the final set, {'묻다', '물어보다', '물었다', '물어보기', '물어'}, conveys the meaning 'ask'. Using these word sets, we compared three different types of embeddings: those derived from subcharacter embeddings in **SCRIPT**, subword embeddings from the PLM, and fusion embeddings augmented by **SCRIPT**.

J Computational Efficiency

J.1 Computational Complexity

As shown in Table 14, we quantify the overhead of each approach by breaking computation into three components: the embedding layer, the Transformer stack, and any model-specific layers.

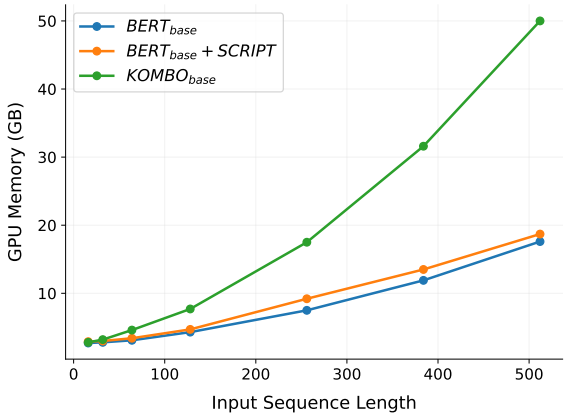
In the embedding layer, standard BERT performs a simple embedding lookup and linear projection, which has constant time complexity with respect to sequence length. Adding **SCRIPT** introduces modest overhead: it applies a GRU-based encoder to contextualize the subcharacter sequence for each token and uses cross-attention to fuse this information with the original subword embedding. Both components are single-layer operations, in contrast to the deep Transformer stack with more than 10 layers. Moreover, **SCRIPT** compresses the subcharacter sequence before cross-attention, keeping sequence lengths short during the expensive fusion step. By comparison, KOMBO's embedding stage is significantly heavier (Ns $\ll$ Nj). It processes the full subcharacter sequence with a GRU, three self-attention layers, and another GRU for compression, making its embedding computation far more costly than **SCRIPT**. In short, both methods add overhead beyond the base model, but **SCRIPT** is much lighter due to its efficient compression and fusion strategy.

In the Transformer stack, **SCRIPT** again aligns closely with the base model. Both the base model (BERT) and BERT+**SCRIPT** operate at the subword level throughout the Transformer layers. This means their self-attention complexity scales with the subword sequence length ($O(N_s{}^2 D)$ per layer, where N_s is the number of subword tokens and D is the hidden dimension), just as in the original model. In contrast, KOMBO converts inputs into much longer character-level sequences (Nc $\gg$ Ns), yielding a per-layer complexity of $O(Nc^2 D)$. This makes KOMBO's Transformer blocks slower and more memory-intensive for the same input.

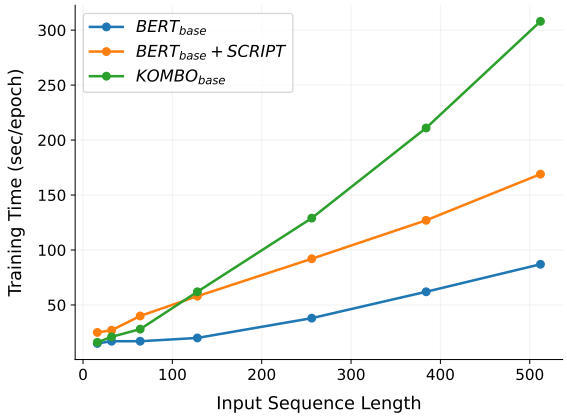
Additionally, KOMBO requires a restoration layer after the Transformer stack to convert character-level outputs back to subword representations, implemented with another GRU. Neither BERT nor BERT+**SCRIPT** requires such steps. Thus, aside from a small embedding-stage overhead, BERT+**SCRIPT** preserves the base model's computational profile, whereas KOMBO incurs substantial extra cost in both the Transformer and output stages.

J.2 Computational Cost

To assess how the computational complexity discussed in the previous section translates into actual computing cost, we empirically evaluate the architectural differences in terms of GPU memory usage and training time.



(a)



(b)

Figure 9: Comparison of computational costs among the base model $\text{BERT}_{\text{base}}$, $\text{BERT}_{\text{base}}$ with **SCRIPT**, and previous Jamo-based PLM, $\text{KOMBO}^{\text{Jamo}}_{\text{base}}$. The results were obtained on the KB-HellaSwag benchmark, a representative Korean NLU task, using a single NVIDIA RTX 3090 GPU. (a) Peak GPU memory usage during training with varying input sequence lengths. (b) Training time per epoch with varying input sequence lengths.

GPU Memory. Figure 9(a) summarizes the resource footprint of each model across varying input sequence lengths. The GPU memory consumption of $\text{BERT}_{\text{base}}$+**SCRIPT** is nearly identical to that of $\text{BERT}_{\text{base}}$ alone, and it remains far lower than that of $\text{KOMBO}_{\text{base}}$, especially for longer sequences. For instance, at an input length of 256 tokens, the $\text{BERT}_{\text{base}}$ model uses about 7.5 GB of GPU

Model	Embedding Layer	Transformer Stacks	Restoration Layer
BERT	$O(1)$	$O({N_s}^2 D)$	-
BERT + **SCRIPT**	$O(N_j D^2 + {N_s}^2 D)$	$O({N_s}^2 D)$	-
KOMBO	$O(N_j D^2 + {N_j}^2 D)$	$O({N_c}^2 D)$	$O(N_c D^2)$

Table 14: Comparison of the computational complexities across the three components of the model's architecture. N_j is the length of subcharacter sequence, N_s is the length of subword sequence, N_c is the length of character sequence, and D is the hidden size.

memory during training, and $\text{BERT}_{\text{base}}$+**SCRIPT** requires approximately 9.2 GB, a relatively small 1.7 GB increase. In contrast, $\text{KOMBO}_{\text{base}}$ at the same sequence length demands roughly 17.5 GB - more than double the memory of the base model. This gap widens with longer inputs: at 512 tokens, $\text{BERT}_{\text{base}}$+**SCRIPT** uses 18.7 GB vs. 17.6 GB for BERT (only a 6% increase), whereas $\text{KOMBO}_{\text{base}}$ soars to about 50 GB, nearly three times the base model's requirement. These results confirm that plug-in design of **SCRIPT** adds minimal memory overhead, while KOMBO's character-level processing and extra layers drastically inflate memory usage for large inputs.

Training Time per Epoch. A similar pattern is observed in training time. As shown in Figure 9(b), **SCRIPT** introduces only a moderate slowdown relative to the base model, whereas $\text{KOMBO}_{\text{base}}$ dramatically reduces training speed as sequence length grows. For a moderate input length (128 tokens), $\text{BERT}_{\text{base}}$+**SCRIPT** requires roughly 58 s per training step compared to 20 s for $\text{BERT}_{\text{base}}$ (about 2.9× slower), and $\text{KOMBO}_{\text{base}}$ takes around 62 s (about 3.1× slower than base). However, as the sequence length increases, KOMBO's runtime cost grows much more rapidly. At 512 tokens, $\text{BERT}_{\text{base}}$+**SCRIPT** processes a batch in roughly 169 s (less than 2× the 87 s required by $\text{BERT}_{\text{base}}$), whereas $\text{KOMBO}_{\text{base}}$ requires about 308 s – over 3.5× the base model's time. This steep slowdown for KOMBO is a direct consequence of operating over a much longer sequence with additional transformation layers, as discussed above. In contrast, **SCRIPT** maintains a moderate runtime overhead, less than 2× the base model even at maximum sequence lengths, making it far more practical than KOMBO in real-world training scenarios.

We note that these trends hold for both encoder-based and decoder-based architectures. In our experiments with the decoder-only KoGPT2 model, adding **SCRIPT** incurred similar slowdowns and only minor memory increases, underscoring the general applicability of **SCRIPT** across model types.

In summary, **SCRIPT** offers a significantly more efficient and practical solution for incorporating subcharacter information than another subcharacter-based approach, such as KOMBO. By sidestepping expensive architectural changes and pre-training requirements, **SCRIPT** maintains almost the same training footprint as the underlying base model in terms of memory. The small overhead introduced by **SCRIPT** is significantly outweighed by its benefits, and it stands in stark contrast to the heavy computational cost of KOMBO. This efficiency makes **SCRIPT** a highly practical plug-and-play module for real-world deployment on large-scale models and datasets. Next, we examine another aspect of training efficiency, the convergence speed of each model during training, to further assess the practical advantages of **SCRIPT**.

J.3 Training Efficiency

In Section J.2, we analyzed the structural efficiency of the proposed method **SCRIPT** compared to another off-the-shelf subcharacter-based model, KOMBO. In this section, we further investigate training efficiency, focusing on the convergence speed across three different models, such as $\text{BERT}_{\text{base}}$ as the base model, $\text{BERT}_{\text{base}}$ + **SCRIPT**, and $\text{KOMBO}_{\text{base}}$, during fine-tuning. Figure 10 compares the convergence points of each model across the nine principal Korean NLU tasks introduced in Section 4.2. As a result, $\text{BERT}_{\text{base}}$+**SCRIPT** consistently converges faster as both $\text{BERT}_{\text{base}}$ and $\text{KOMBO}_{\text{base}}$, achieving superior performance with fewer training epochs. Specifically, it converged faster on six out of nine tasks, and matched the convergence speed on the remaining three tasks. These results demonstrate that our method significantly enhances learning efficiency. Furthermore, though **SCRIPT** leverages both subword and subcharacter embeddings, it not only outperforms the model utilizing solely

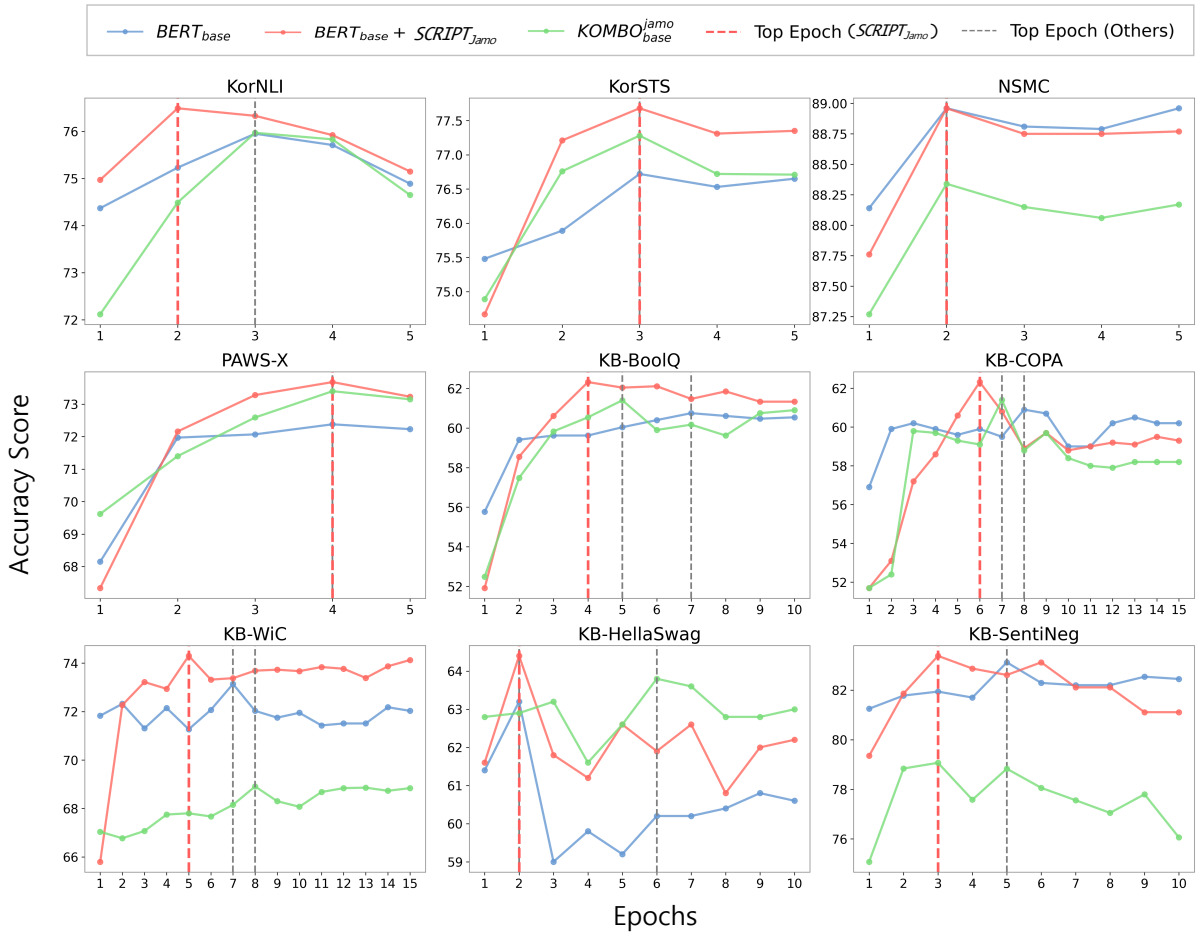


Figure 10: The graphs show the fine-tuning performances of three models, $\text{BERT}_{\text{base}}$, $\text{BERT}_{\text{base}}$+**SCRIPT**, and $\text{KOMBO}_{\text{base}}$, across nine NLU tasks. The x-axis represents the number of epochs for each task, and the y-axis indicates the accuracy score. Two types of dotted vertical lines are overlaid on the line graphs for each model: the red dotted line marks the epoch at which the model with the proposed **SCRIPT** module achieves its best performance, while the gray dotted lines indicate the best-performing epochs for the other two models.

32k subwords ($\text{BERT}_{\text{base}}$) and fewer than 200 subcharacters ($\text{KOMBO}_{\text{base}}$) but also converges more rapidly, highlighting its strong adaptation to Korean language understanding.

Overall, our comprehensive evaluation demonstrates that **SCRIPT** offers a substantially more efficient and scalable alternative to prior subcharacter-based approaches. By injecting subcharacter compositional knowledge directly into existing PLM embeddings, **SCRIPT** enriches the model's representational capacity while preserving the computational profile of the base model. This dual advantage, greater linguistic expressiveness with only marginal computational overhead, establishes **SCRIPT** as a practical and robust solution for real-world deployment in Korean NLP.