# Morpheme Matters: Morpheme-Based Subword Tokenization for Korean Language Models

**Donghyeok Lee Jeongyeon Park Kyungbeen Cho Jae Sung Lee***
Language Knowledge Engineering Lab, Chungbuk National University
{nlp_dohy,parkjeongyeon,kyungbeen,jasonlee}@cbnu.ac.kr

## Abstract

Tokenization plays a crucial role in the performance of language models. However, most existing tokenizers rely on frequency-based segmentation, which fails to capture the morphological structure of languages and often leads to inefficient token representations. In this study, we propose a novel tokenization method that emphasizes the importance of Korean morphological structures in *eojeol* (Korean spacing unit). This method is designed to accommodate both inter-*eojeol* segmentation and intra-*eojeol* segmentation, enabling the selection of subwords based on morphemes. We pretrained a language model using the proposed method and evaluated its performance on Korean benchmark tasks. Experimental results demonstrate that the proposed method generally outperforms existing approaches. Notably, it produces significantly fewer tokens per input sequence, indicating its effectiveness and efficiency for Korean language modeling. The code is available at https://github.com/Dohy-Lee/mob.

## 1 Introduction

Tokenizers play a crucial role in language model performance, as their token construction directly affects how linguistic information is represented. Common subword tokenization methods, such as Byte-Pair Encoding (BPE; Gage, 1994; Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012), are unsupervised and language-agnostic approaches based on subword frequency. However, these frequency-driven methods often yield inefficient tokens that fail to reflect the morphological structure of words, leading to over-segmentation and a increase in the number of token instances (Klein and Tsarfaty, 2020; Bostrom and Durrett, 2020). Such inefficiencies have been shown to detrimentally impact language model performance (Park et al., 2021; Schmidt et al., 2024; Goldman et al., 2024).

To mitigate this issue, recent studies have explored morphology-aware tokenization. Strategies include building vocabularies with morphological information (Hofmann et al., 2021), preserving morphological boundaries during tokenization (Hofmann et al., 2022), decomposing compounds for morpheme alignment (Minixhofer et al., 2023), and constraining subword formation to avoid crossing morpheme boundaries (Bauwens and Delobelle, 2024). Such morphology-sensitive methods have shown effectiveness in multilingual and morphologically rich language settings (Chen and Fazio, 2021; Mager et al., 2022; Toraman et al., 2023).

Korean is a morphologically rich language that is both agglutinative and inflectional (Sohn, 1999; Yeon and Brown, 2019). A Korean sentence consists of *eojeol*s. An *eojeol* is the basic orthographic spacing unit, consisting of one or more agglutinated morphemes. It does not directly correspond to the notion of a "word" in English, as grammatical morphemes such as case particles and verbal endings are typically attached to lexical stems within an *eojeol* (Song and Park, 2019).[1] Moreover, a morpheme can surface in multiple inflected forms, and two morphemes may fuse together, making simple surface segmentation insufficient for morphological analysis (Matteson et al., 2018; Jo et al., 2023).

When the morphological structure of Korean *eojeol*s is taken into account, token generation efficiency improves notably. A representative approach in this line is Morpheme-aware Subword

*Corresponding Author

[1]For instance, consider "공부했겠다" (would have studied). While this expression requires three distinct words in English, it constitutes a single *eojeol* in Korean. This unit is an agglutination of the stem "공부하-" (study) with multiple functional morphemes: "-았-" (past tense), "-겠-" (modal/conjecture), and "-다" (ending). During agglutination, the stem "하-" and the past tense morpheme "-았-" are phonologically fused, resulting in the contracted form "했-".



*Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics*
*Volume 2: Short Papers*, pages 297–306

Tokenization (MoA; Park et al., 2020). MoA begins by applying a morphological analyzer to segment sentences into surface form units that respect morpheme boundaries. During this process, morpheme boundaries are approximated using spaces, and a special *eojeol*-boundary token is introduced to mark divisions between *eojeol*s. SentencePiece (Kudo and Richardson, 2018) is then applied to generate tokens. While the explicit boundary token effectively preserves *eojeol* boundaries, it also increases the number of input tokens in proportion to the number of *eojeol*s in a sentence, which can potentially affect language model performance (Levy et al., 2024). Furthermore, since MoA operates only on surface forms without restoring morphemes to base forms, different inflected variants of the same morpheme may be represented as distinct tokens.

To address these limitations, we introduce Morpheme-based Subword Tokenization (MoB), a method that integrates morpheme restoration with *eojeol* structure. In MoB, morphemes are restored to their canonical base forms during segmentation, after which subword tokenization is applied over these restored units. Unlike the previous approach, which relies on explicit delimiter tokens, MoB implicitly encodes both *eojeol* and morpheme boundary information within the token structure. This design allows the model to capture structural boundaries more effectively while reducing the overall number of tokens and ensuring that inflectional variants of the same morpheme are represented consistently.

To evaluate the efficiency and effectiveness of the proposed method, we pretrained a Korean BERT language model using MoB tokenization and assessed its performance on a range of Korean Natural Language Understanding (NLU) tasks. The experimental results demonstrate that the proposed method reduced the average number of tokens and improved task performance compared to the baseline model (MoA), thereby validating its efficiency and effectiveness.

## 2 Morpheme-based Subword Tokenization

This study proposes Morpheme-based Subword Tokenization (MoB) as an enhanced version of Morpheme-aware Subword Tokenization (MoA), with two key improvements:

1. It performs complete morpheme segmentation on base forms, enabling consistent tokenization of morphological variants.

2. It implicitly encodes both *eojeol* and morpheme boundary information within the token structure, which significantly reduces the total number of tokens per input sequence caused by *eojeol* boundary token.

The MoB process begins with the morphological analysis of a sentence and utilizes morphemes of base forms. Each *eojeol* boundary is marked by attaching an inter-*eojeol* prefix (‘_’) before the first morpheme of the corresponding *eojeol*. Likewise, morpheme boundaries within each *eojeol* are marked using an intra-*eojeol* prefix (‘+’) attached to the initial character of each morpheme except the first morpheme in the *eojeol*.[2] These preprocessed characters are then split into character-level unigrams, which serve as the initial token sequence. Importantly, the prefixes are tightly bound to the following character and are not counted as separate characters within a token.

We follow a standard BPE-based procedure inspired by SentencePiece (Kudo and Richardson, 2018) for token vocabulary construction and tokenization, with an additional constraint to account for two types of *eojeol* prefixes. Specifically, during the merging of unigrams into bigrams, a unigram annotated with an *eojeol* prefix is permitted only in the first position of a bigram and is disallowed in the second position. The algorithm for token vocabulary construction is provided in Appendix A.1.

Table 1 presents an illustrative example to highlight the differences among the BPE, MoA, and MoB tokenization methods. That is, each tokenization algorithm is applied to a given small corpus, and the resulting tokenized sentences are presented. In the BPE result, the sequence “교가” (school song) is merged as a frequent bigram that appears in both “학교가” and “등교가”. However, this merged unit crosses morpheme boundaries—combining part of noun “학교” (school), “등교” (going to school) with a case particle (“가”)—and thus forms an undesirable token. In contrast, the MoA method explicitly marks morpheme boundaries using the morpheme-segment prefix “▁”,

[2] In this paper, the prefixes are denoted as (‘_’) and (‘+’) for illustrative purposes; however, the actual implementation employs rarely used characters. The inter-*eojeol* and intra-*eojeol* prefixes are implemented as U+2581 and U+29FE, respectively.

| Tokenizer | Tokenized sequence | Token count |
|---|---|---|
| Corpus | 즐거운␣학교가␣있어␣등교가␣즐겁다<br>(There is a joyful school, so going to school is joyful) | |
| BPE | _즐 / 거 / 운 / _학 / **교가** / _있 / 어 / _등 / **교가** / _즐 / 겁 / 다 | 12 |
| MoA | ▁즐 / 거 / 운 / * / ▁학 / 교 / ▁가 / * / ▁있 / ▁어 / * / ▁등 / 교 / ▁가 / * / ▁즐 / 겁 / ▁다 | 18 |
| MoB | _**즐겁** / +ㄴ / _학 / 교 / +가 / _있 / +어 / _등 / 교 / +가 / _**즐겁** / +다 | 12 |

Table 1: Tokenization examples. Symbols: _ = inter-*eojeol* prefix, + = intra-*eojeol* prefix, ▁ = morpheme-segment prefix, * = *eojeol* boundary token, / = token delimiter.

| | |
|---|---|
| **Input sentence** | 즐거운␣학교가␣있어␣등교가␣즐겁다<br>(There is a joyful school, so going to school is joyful) |
| **Output: surface form analysis** | 즐거운␣학교+가␣있+어␣등교+가␣즐겁+다 |
| **Output: base form analysis** | 즐겁+ㄴ␣학교+가␣있+어␣등교+가␣즐겁+다 |

Table 2: Two types of MeCab-ko analysis results. At the surface form level, '즐거운' is analyzed as '즐거운', whereas at the base form level it becomes '즐겁+ㄴ'. Here, the symbol '+' is a virtual marker used to indicate morpheme boundaries within an *eojeol*; it does not appear in the actual output.

thereby preventing such cross-morpheme bigrams from being formed. MoB not only prevents cross-morpheme merges in the same way as MoA, but also merges bigrams whose constituent morphemes share the same base forms. For instance, "즐거운" (joyful) and "즐겁다" (is joyful) both share the same base form of morpheme "즐겁". Since this morpheme appears twice in the input text, it is merged into a bigram token accordingly.

In terms of the number of generated tokens, MoB is more efficient than MoA, producing only 12 tokens compared to MoA's 18. Notably, MoB successfully identifies and includes a desirable bigram, "즐겁", and empirical results further demonstrate its superior performance. Although BPE also generates 12 tokens, it includes an undesirable bigram, "교가" which may hurt the performance. In fact, previous studies have reported that the MoA method outperforms BPE in general (Park et al., 2020).

## 3 Experiments

**Preprocessing** For fair comparison, both MoA and MoB methods utilized MeCab-ko[3] morphological analyzer. The MeCab-ko supports analysis on both the surface level and base level, as shown in Table 2. For the pretraining data, we filtered out non-Korean text and converted Sino-Korean words written in Chinese characters into their Korean pronunciations. We then constructed a 32k-sized token vocabulary from a Korean Wikipedia dump (780MB)[4] and a subset (3.7GB) of the AI-Hub corpus[5], which were also used as the pretraining dataset.

**Pretraining** We adopted BERT (Devlin et al., 2019) as the base model, using a publicly available implementation[6]. For pretraining, we chose $\text{BERT}_{medium}$ with 41M parameters (Turc et al., 2019) and trained it for up to 1M steps on four NVIDIA A6000 48GB GPUs over approximately two weeks.

**Evaluation** We evaluated our method using two approaches: extrinsic evaluation on downstream tasks and intrinsic evaluation of token compression, measured by the numbers of token instances and token types as indicators of token quality (Goldman et al., 2024). We used five downstream fine-tuning tasks in Korean: Machine Reading Comprehension using KorQuAD (Lim et al., 2019), Semantic Textual Similarity (STS) using KorSTS (Ham et al., 2020), Natural Language Inference (NLI) using KorNLI (Ham et al., 2020), Sentiment Analysis using NSMC [7], and Paraphrase Identification using the Korean subset of PAWS-X (Yang et al., 2019).

Statistics of the fine-tuning datasets as well as the hyperparameters used for pretraining and fine-tuning are provided in Appendix A.3.

## 4 Results and Discussion

**Fine-tuning performance** Table 3 presents the fine-tuning performance on the five downstream

[3] https://bitbucket.org/eunjeon/mecab-ko.
[4] https://dumps.wikimedia.org/kowiki/
[5] https://www.aihub.or.kr/
[6] https://github.com/codertimo/BERT-pytorch
[7] Naver Sentiment Movie Corpus for sentimental analysis, https://github.com/e9t/nsmc

| | KorQuAD | | KorSTS | KorNLI | NSMC | PAWS-X |
|---|---|---|---|---|---|---|
| Tokenizer | EM | F1 | $\rho$ | Acc | Acc | Acc |
| BPE | 49.72 ±0.31 | 65.07 ±0.35 | 63.00 ±0.77 | 64.47 ±0.39 | 85.04 ±0.04 | 61.11 ±0.82 |
| MoA | 50.25 ±0.57 | 66.35 ±0.46 | 68.30 ±0.74 | 67.32 ±0.18 | **86.07 ±0.08** | 59.20 ±2.57 |
| MoB | **51.65 ±0.38** | **69.33 ±0.14** | **69.53 ±0.60** | **68.61 ±0.22** | 85.73 ±0.07 | **62.66 ±0.89** |

Table 3: Performance of BPE, MoA, and MoB tokenizers. Values are mean ± standard deviation. KorQuAD is evaluated on the dev set due to the unavailability of a public test set, while KorSTS, KorNLI, NSMC, and PAWS-X are evaluated on their respective test sets. KorSTS is reported as Spearman correlation ($\rho \times 100$), and KorNLI, NSMC, and PAWS-X are reported as Accuracy (%). Please refer to Table 10 in Appendix A.3 for the statistical significance of the comparison between MoA and MoB.

| Dataset | Number of Token Types | | | Number of Token Instances | | |
|---|---|---|---|---|---|---|
| | MoA | MoB | $\Delta$ | MoA | MoB | $\Delta$ |
| KorQuAD | 14,551 | 13,183 | **91%** | 430.53 | 340.13 | **79%** |
| KorSTS | 5,795 | 5,499 | **95%** | 52.26 | 41.00 | **78%** |
| KorNLI | 5,592 | 5,159 | **92%** | 60.76 | 48.18 | **79%** |
| NSMC | 10,445 | 10,377 | **99%** | 28.18 | 23.97 | **85%** |
| PAWS-X | 6,391 | 5,901 | **92%** | 90.60 | 77.35 | **85%** |
| Average | 8,555 | 8,024 | **94%** | 132.07 | 106.13 | **81%** |

Table 4: Comparison of the size of token types and instances. The number of token types denotes the total number of unique tokens observed in the dataset, and the number of token instances denotes the average frequency with which a given token appears within a single input sequence. $\Delta$ is calculated as (MoB/MoA) $\times$ 100.

tasks. MoB outperformed MoA on all tasks, with the exception of NSMC. For NSMC, MoB method showed slightly lower performance than MoA. The NSMC dataset showed a different trend, which we attribute to its nature as an online corpus containing numerous neologisms, abbreviations, and typographical errors, making it difficult for the morphological analyzer to produce accurate analyses.

**Token size** Table 4 summarizes the number of unique token types and the average number of tokens per input sequence generated by each tokenization method. MoB yielded on average 6 percentage fewer token types than MoA across all development datasets. This reduction in token variety indicates that MoB achieves more efficient vocabulary compression by grouping morphologically different but semantically equivalent word forms into a single token through morpheme restoration.

Furthermore, the average number of token instances per input sequence was about 19 percentage lower for MoB than for MoA across all tasks. This outcome can be attributed to MoB's design, which avoids inserting an *eojeol* boundary token. Instead, MoB employs the inter-*eojeol* prefix and intra-*eojeol* prefix to denote both *eojeol* and morpheme boundaries directly within existing tokens.

In general, introducing one prefix marker can potentially double the number of unigram token types, since each character may appear in both prefixed and unprefixed forms. With two types of prefix markers, the theoretical maximum increase in unigram types could be up to threefold. However, in practice, the use of such prefix markers did not lead to a substantial increase in the total number of tokens. On the contrary, the use of the inter-*eojeol* prefix (along with the intra-*eojeol* prefix), rather than the *eojeol*-boundary token, appears to have contributed to more compact and efficient tokenization.

**Ablation study** Table 5 presents the results of an ablation study conducted to examine the effects of the intra-*eojeol* prefix and morpheme restoration. Using the Full Model as the reference, removing the intra-*eojeol* prefix results in performance decreases on most tasks, while performance on NSMC remains unchanged. When only the morpheme restoration feature is removed, performance drops are observed across all tasks. When both features are removed simultaneously, performance decreases become larger on most tasks, whereas NSMC shows a slight performance improvement. The NSMC evaluation results seem to be affected by characteristics of the NSMC dataset (see Appendix A.2). Overall, these results indicate that the intra-*eojeol* prefix and morpheme restoration have different effects depending on the task, while generally contributing to improved performance on most benchmarks.

| Variant | KorQuAD (F1) | | KorSTS ($\rho$) | | KorNLI (Acc) | | NSMC (Acc) | | PAWS-X (Acc) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Score | Δ | Score | Δ | Score | Δ | Score | Δ | Score | Δ |
| Full Model | **69.14** | – | **70.32** | – | **68.84** | – | 85.84 | – | **63.15** | – |
| – w/o IntraP | 67.41 | –1.73 | 70.01 | –0.31 | 68.10 | –0.74 | 85.84 | 0.00 | 61.65 | –1.50 |
| – w/o MRest | 67.34 | –1.80 | 68.86 | –1.46 | 67.90 | –0.94 | 85.70 | –0.14 | 61.60 | –1.55 |
| – w/o IntraP & MRest | 66.14 | –3.00 | 67.68 | –2.64 | 67.21 | –1.63 | **85.94** | +0.10 | 56.95 | –6.20 |

Table 5: Ablation study on Intra-*eojeol* Prefix (IntraP) and Morpheme Restoration (MRest). The full model includes both IntraP and MRest. Δ indicates performance change relative to the full model. $\rho$ denotes Spearman Correlation $\times$ 100, and Acc denotes Accuracy (%).

## 5 Related Works

Park et al. (2020) proposed the Morpheme-aware Subword tokenization (MoA) method, which is adopted as a baseline in this paper. Reflecting the characteristics of the Korean writing system, they explored a wide range of tokenization strategies by segmenting *eojeol*s into consonants and vowels, syllables, morphemes, subwords, morpheme-aware subwords (MoA), and full words (*eojeol*s). Among these, MoA achieved the best performance. As described earlier in this paper, MoA explicitly distinguishes morpheme boundaries in surface forms and employs a special token to mark *eojeol* boundaries.

Lee et al. (2024) addressed the over-segmentation problem of BPE, in which compound words are excessively split and their original meanings are distorted, by populating a fixed proportion of the token vocabulary with high-frequency long tokens. Although this strategy was additionally applied to MoA, the resulting performance improvements were marginal.

Morphology-based tokenization methods in languages other than Korean have mainly been studied for morphologically rich languages. Nzeyimana and Niyongabo Rubungo (2022) implemented a Rwandan encoder language model using a two-stage process. In the first stage, morphological analysis results are encoded, and used as tokens for the second-stage language model encoder. This approach differs fundamentally from conventional vocabulary-based tokenization methods.

Asgari et al. (2025) applied a morpheme-boundary-aware tokenization scheme to English, Russian, Hungarian, and Arabic. Their approach takes morphologically segmented input as a pre-processing step; however, the paper does not detail how word and morpheme boundaries are distinguished within tokens.

Bayram et al. (2025) proposed a tokenization method for Turkish that performs morpheme segmentation based on a dictionary, while applying BPE to out-of-vocabulary words. To reduce vocabulary size, phonological normalization was applied, and special whitespace tokens were introduced to represent word boundaries.

Overall, prior studies primarily focus on handling morpheme boundaries in surface forms and introduce additional special tokens to represent word (or *eojeol*) boundaries. Such designs tend to increase vocabulary size and reduce token representation efficiency. In contrast, the tokenization method proposed in this paper, which is based on restored base-form morphemes and employs explicit *eojeol*- and morpheme-level prefixes, addresses these limitations and improves tokenization efficiency.

## 6 Conclusion

Tokenization plays a critical role in the performance of language models, and recent research has focused on developing methods that can handle a wide range of domains and languages. In this study, we propose Morpheme-based Subword Tokenization (MoB), a tokenization method that incorporates the linguistic characteristics of Korean. The proposed method restores base forms of morpheme to facilitate the recognition of tokens sharing the same base form. In addition, it reflects the structure of Korean *eojeol*s by introducing two types of prefixes: one for word (*eojeol*) boundaries by inter-*eojeol* prefix and another for morpheme boundaries by intra-*eojeol* prefix.

We pretrained a BERT-based language model using this tokenization method and evaluated it on five fine-tuning tasks. Despite minor performance degradation in domain-specific datasets with frequent morphological analysis errors, the proposed method proved more effective overall, achieving higher performance while reducing the number of generated tokens by about 19% compared to the baseline method.

## Limitations

This study evaluates the proposed method under limited training configurations of a BERT-based language model. A broader exploration of model architectures and training settings is required for a more comprehensive assessment. In addition, although the model was evaluated on five fine-tuning tasks, further validation on other task types remains for future work.

Further studies are needed to compare our approach with alternative paradigms, including dynamic tokenization (Feher et al., 2025) and tokenizer-free modeling approaches spanning character-level, subword-inductive, and byte-level hierarchical representations (Pagnoni et al., 2025; Tay et al., 2022; Clark et al., 2022; Yu et al., 2023; Wang et al., 2024). Such comparisons would clarify the trade-offs between explicit structural constraints and fully dynamic or implicit representations in terms of efficiency, robustness, and downstream performance.

For fair comparison with prior work, we relied on the MeCab-ko morphological analyzer. However, performance may vary with different analyzers, particularly those more robust to morphological analysis errors, which warrants further investigation.

While the method is specifically designed for Korean, its applicability to other morphologically rich or agglutinative languages has not been examined and remains an open research question.

Finally, applying the proposed tokenization to generative models requires converting morpheme base forms back into surface forms. Although this is expected to be handled via relatively simple post-processing, its effectiveness has not been empirically validated and should be explored in future work.

## Acknowledgements

We are grateful to the anonymous reviewers for their thorough and insightful reviews, which helped improve this paper.

## References

Ehsaneddin Asgari, Yassine El Kheir, and Mohammad Ali Sadraei Javaheri. 2025. Morphbpe: A morpho-aware tokenizer bridging linguistic complexity for efficient llm training across morphologies. *Preprint*, arXiv:2502.00894.

Thomas Bauwens and Pieter Delobelle. 2024. BPE-knockout: Pruning pre-existing BPE tokenisers with backwards-compatible morphological semi-supervision. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5810–5832, Mexico City, Mexico. Association for Computational Linguistics.

M. Ali Bayram, Ali Arda Fincan, Ahmet Semih Gümüş, Sercan Karakaş, Banu Diri, Savaş Yıldırım, and Demircan Çelik. 2025. Tokens with meaning: A hybrid tokenization approach for nlp. *Preprint*, arXiv:2508.14292.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

William Chen and Brett Fazio. 2021. Morphologically-guided segmentation for translation of agglutinative low-resource languages. In *Proceedings of the 4th Workshop on Technologies for MT of Low Resource Languages (LoResMT2021)*, pages 20–31, Virtual. Association for Machine Translation in the Americas.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

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.

Darius Feher, Ivan Vulić, and Benjamin Minixhofer. 2025. Retrofitting large language models with dynamic tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29866–29883, Vienna, Austria. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. 2024. Unpacking tokenization: Evaluating text compression and its correlation with model performance. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2274–2286, Bangkok, Thailand. Association for Computational Linguistics.

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.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Derivational morphology improves BERT's interpretation of complex words. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3594–3608, Online. Association for Computational Linguistics.

Valentin Hofmann, Hinrich Schuetze, and Janet Pierrehumbert. 2022. An embarrassingly simple method to mitigate undesirable properties of pretrained language model tokenizers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–393, Dublin, Ireland. Association for Computational Linguistics.

Eunkyul Leah Jo, Kyuwon Kim, Xihan Wu, KyungTae Lim, Jungyeul Park, and Chulwoo Park. 2023. K-UniMorph: Korean Universal Morphology and its feature schema. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6613–6623, Toronto, Canada. Association for Computational Linguistics.

Stav Klein and Reut Tsarfaty. 2020. Getting the ##life out of living: How adequate are word-pieces for modelling complex morphology? In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 204–209, Online. Association for Computational Linguistics.

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

Jungseob Lee, Hyeonseok Moon, Seungjun Lee, Chanjun Park, Sugyeong Eo, Hyunwoong Ko, Jaehyung Seo, Seungyoon Lee, and Heuiseok Lim. 2024. Length-aware byte pair encoding for mitigating oversegmentation in Korean machine translation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2287–2303, Bangkok, Thailand. Association for Computational Linguistics.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15339–15353, Bangkok, Thailand. Association for Computational Linguistics.

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

Manuel Mager, Arturo Oncevay, Elisabeth Mager, Katharina Kann, and Thang Vu. 2022. BPE vs. morphological segmentation: A case study on machine translation of four polysynthetic languages. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 961–971, Dublin, Ireland. Association for Computational Linguistics.

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.

Benjamin Minixhofer, Jonas Pfeiffer, and Ivan Vulić. 2023. CompoundPiece: Evaluating and improving decompounding performance of language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 343–359, Singapore. Association for Computational Linguistics.

Antoine Nzeyimana and Andre Niyongabo Rubungo. 2022. KinyaBERT: a morphology-aware Kinyarwanda language model. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5347–5363, Dublin, Ireland. Association for Computational Linguistics.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. 2025. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258, Vienna, Austria. Association for Computational Linguistics.

Hyunji Hayley Park, Katherine J. Zhang, Coleman Haley, Kenneth Steimel, Han Liu, and Lane Schwartz. 2021. Morphology matters: A multilingual language modeling analysis. *Transactions of the Association for Computational Linguistics*, 9:261–276.

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.

Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. In *Proceedings of the 2024 Conference on*

*Empirical Methods in Natural Language Processing*, pages 678–702, Miami, Florida, USA. Association for Computational Linguistics.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Homin Sohn. 1999. *The Korean Language*. Cambridge University Press, Cambridge, UK.

Hyun-Je Song and Seong-Bae Park. 2019. Korean morphological analysis with tied sequence-to-sequence multi-task model. 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 1436–1441, Hong Kong, China. Association for Computational Linguistics.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*.

Cagri Toraman, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozcelik. 2023. Impact of tokenization on language models: An analysis for turkish. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 22(4).

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Junxiong Wang, Tushaar Gangavarapu, Jing Nathan Yan, and Alexander M Rush. 2024. Mambabyte: Token-free selective state space model. In *First Conference on Language Modeling*.

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. 2019. *Korean: A Comprehensive Grammar*, 2nd edition. Routledge, London, UK.

Lili Yu, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. Megabyte: Predicting million-byte sequences with multiscale transformers. In *Advances in Neural Information Processing Systems*, volume 36, pages 78808–78823. Curran Associates, Inc.

# A Appendix

## A.1 MoB Vocabulary Construction

The MoB vocabulary construction algorithm takes as input unigrams annotated with inter-*eojeol* and intra-*eojeol* prefixes and outputs a token vocabulary consisting of $k$ entries. The procedure is presented in pseudocode in Algorithm 1 and operates as follows. Given a corpus $C$ represented as a sequence of initial tokens, all unique tokens are first collected to form the initial vocabulary $V$ (Line 3). The algorithm then identifies the most frequent bigram pair $(t_l, t_r)$ in $C$. If $t_r$ begins with a boundary prefix (‘_’ or ‘+’), the algorithm skips this pair and continues searching for the next most frequent valid bigram (Lines 5–8). Otherwise, the selected bigram is merged into a new token and added to the vocabulary $V$ (Lines 10–11). The corpus $C$ is then updated to reflect the newly created token (Line 12), and the process repeats until the vocabulary size reaches $k$.

**Algorithm 1** Vocabulary Construction for Morpheme-based Subword Tokenization

1: Input: Unigram set of corpus $C$, marked with inter-*eojeol* / intra-*eojeol* prefixes ( ‘_’/‘+’)
2: Output: Vocabulary set $V$ ($|V| \leq$ k)
3: Initialize $V \leftarrow$ all unique tokens $t$ in $C$
4: **while** $|V| < k$ **do**
5: Find the most frequent bigram $(t_l, t_r)$ in $C$
6: **if** no more bigram candidate **then**
7: **break**
8: **end if**
9: **if** $t_r[0] =$ ‘+’ *or* ‘_’ **then**
10: Skip this bigram
11: **continue**
12: **end if**
13: $t_{\text{new}} \leftarrow t_l + t_r$
14: $V \leftarrow V \cup \{t_{\text{new}}\}$
15: Update $C$ with new $V$
16: **end while**
17: **return** $V$

## A.2 Error Analysis in NSMC

NSMC dataset, as an online movie review dataset, contains numerous neologisms, abbreviations, and

| | |
|---|---|
| Raw Text | 간만에 ␣볼만한 ␣영화<br>( A movie worth watching after a long time ) |
| MoA | ▁간 / ▁만 / ▁에 / * / ▁볼 / ▁만 / ▁한 / * / ▁영화 |
| MoB | _가 / +ㄴ / +만 / +에 / _보 / +ㄹ / +만 / +하 / +ㄴ / _영화 |
| Expected MoB | _간만 / +에 / _보 / +ㄹ / +만 / +하 / +ㄴ / _영화 |

(a) Errors caused by omission.

| | |
|---|---|
| Raw Text | 넘 ␣잼나게 ␣잘 ␣봤어요<br>( I had so much fun watching it ) |
| MoA | ▁넘 / * / ▁잼 / ▁나 / ▁게 / * / ▁잘 / * / ▁봤 / ▁어요 |
| MoB | _넘 / _재 / +ㅁ / +나 / +게 / _잘 / _보 / +았 / +어요 |
| Expected MoB | _너무 / _재미나 / +게 / _잘 / _보 / +았 / +어요 |

(b) Errors caused by contraction.

**Symbols:** _ : Inter-*eojeol* prefix, + : Intra-*eojeol* prefix, ▁ : Morpheme-segment prefix, * : *eojeol*-boundary token, / : Token delimiter.

Table 6: Examples of MoB tokenization errors in NSMC task

typographical errors. These forms often lead the morphological analyzer to produce incorrect analyses, yielding misaligned morpheme boundaries or inaccurate base forms, thereby hindering effective token generation.

As shown in Table 6a, the *eojeol* "간만에"—a colloquial contraction of "오래간만에" formed by omitting "오래" ("after a long time")—should have been analyzed as "간만/+에". However, the morphological analyzer returned an incorrect result: "가/+ㄴ/+만/+에". In addition, the NSMC dataset contains many spoken-style expressions that the morphological analyzer failed to recognize correctly. For example, as shown in Table 6b, the colloquial contraction "넘", derived from "너무" meaning "very" or "too," was not properly restored. Similarly, "잼", a slang form of "재미" meaning "fun," was incorrectly analyzed as "재" and "+ㅁ".

### A.3 Dataset Statistics, Hyperparameters, and Statistical Significance Analysis

| Task | Train | Dev | Test |
|---|---|---|---|
| KorQuAD | 60,407 | 5,774 | - |
| KorSTS | 5,749 | 1,500 | 1,379 |
| KorNLI | 942,854 | 2,490 | 5,010 |
| NSMC | 135,000 | 15,000 | 50,000 |
| PAWS-X | 49,410 | 1,965 | 1,972 |

Table 7: Statistics of the fine-tuning datasets. The statistics report the number of sentences for NSMC, question-passage pairs for KorQuAD, and two sentence pairs for KorSTS, KorNLI, and PAWS-X.

| Parameter | Value |
|---|---|
| Number of layers | 8 |
| Hidden size | 512 |
| Attention heads | 8 |
| Batch size | 64 |
| Learning rate | 5e-5 |
| Warm up | 10,000 |
| Max length | 512 |
| Optimizer | AdamW |

Table 8: Pretraining hyperparameters

| Parameter | KorQuAD | KorSTS | KorNLI | NSMC | PAWS-X |
|---|---|---|---|---|---|
| Epoch | 5 | 3 | 5 | 3 | 5 |
| Batch size | 64 | 64 | 64 | 64 | 64 |
| Learning rate | 5e-5 | 1e-4 | 5e-5 | 5e-5 | 1e-4 |
| Dropout | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Warm up | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Max length | 128 | 128 | 128 | 128 | 128 |
| Optimizer | AdamW | AdamW | AdamW | AdamW | AdamW |

Table 9: Fine-tuning hyperparameters

| Task | MoA | MoB | Difference (95% CI) | $p$-value |
|---|---|---|---|---|
| KorQuAD (EM) | $50.25 \pm 0.57$ | $51.65 \pm 0.38$ | +1.40 [1.08, 1.73] | $p < 0.001$ |
| KorQuAD (F1) | $66.35 \pm 0.46$ | $69.33 \pm 0.14$ | +2.98 [2.55, 3.40] | $p < 0.001$ |
| KorSTS | $68.30 \pm 0.74$ | $69.53 \pm 0.60$ | +1.23 [0.15, 2.30] | $p = 0.031$ |
| KorNLI | $67.32 \pm 0.18$ | $68.61 \pm 0.22$ | +1.29 [0.97, 1.60] | $p < 0.001$ |
| NSMC | $86.07 \pm 0.08$ | $85.73 \pm 0.07$ | -0.34 [-0.47, -0.21] | $p < 0.001$ |
| PAWS-X | $59.20 \pm 2.57$ | $62.66 \pm 0.89$ | +3.46 [0.73, 6.19] | $p = 0.021$ |

Table 10: Statistical significance analysis for Table 3. Values are reported as the mean $\pm$ standard deviation over seven runs with different random seeds. The Difference column shows the result of MoB$-$MoA (positive values favor MoB; negative values favor MoA). Brackets $[\cdot, \cdot]$ denote the 95% confidence interval for the mean difference, and $p$-values are from two-sided paired $t$-tests (paired by the random seed).