# Comparative Analysis of Tokenization Methods for Korean Sentiment Analysis: SentencePiece vs. Mecab

**KeeTaek Yang**
AIFFEL online no.9

## Abstract

Tokenization is a critical preprocessing step in natural language processing (NLP), especially for morphologically rich languages like Korean. In this study, I investigate the impact of two distinct tokenization methods, SentencePiece and Mecab, on sentiment analysis performance using the Naver Sentiment Movie Corpus (NSMC). SentencePiece, a subword-based tokenizer, offers flexibility by creating tokens based on statistical frequency, while Mecab, a morpheme-based tokenizer, segments text according to linguistic rules specific to Korean. I implement a bidirectional LSTM model to evaluate each tokenization method, measuring accuracy, precision, and recall. The results indicate that Mecab, with its focus on linguistic morphemes, captures sentiment-bearing nuances more effectively, outperforming SentencePiece in terms of overall accuracy. This analysis provides insights into the strengths and weaknesses of each approach, offering practical recommendations for Korean NLP tasks where sentiment nuances are crucial. Future work includes exploring hybrid tokenization approaches that balance linguistic specificity with computational efficiency.

## 1 Introduction to Tokenization Challenges in Korean Sentiment Analysis

The rapid advancement of natural language processing (NLP) has significantly enhanced our ability to analyze and understand text in multiple languages. However, applying these techniques to Korean, a language with unique morphological and syntactic structures, presents distinct challenges. Korean relies on complex word morphology, where a single word can express nuanced meanings through particles and affixes, and lacks clear word delimiters found in languages like English. Such characteristics pose difficulties for NLP models, making tokenization – the process of dividing text into linguistically meaningful units – a critical pre-processing step in Korean NLP tasks, including sentiment analysis.

Tokenization directly impacts model performance, as different tokenization methods can lead to variations in vocabulary size, representation accuracy, and model interpretability. For sentiment analysis in particular, where the goal is to capture emotional tone and sentiment-bearing expressions, effective tokenization can improve a model's ability to detect subtle sentiment shifts and nuanced expressions. Traditional morpheme-based tokenizers, like Mecab, have been widely used to handle Korean's rich morphology by segmenting words into their constituent morphemes. In contrast, subword methods such as SentencePiece, which are based on data-driven approaches, allow for flexible vocabulary control and address the challenges posed by out-of-vocabulary words. Despite their strengths, the relative effectiveness of these methods in capturing sentiment-bearing units in Korean text remains underexplored.

Written as a practice paper for AIFFEL course. Do not handle as an actual academic paper.

This study aims to address this gap by comparing the performance of SentencePiece and Mecab on a sentiment analysis task using the Naver Sentiment Movie Corpus (NSMC), a popular dataset of Korean movie reviews. By examining both quantitative metrics and qualitative case studies, I seek to understand how each tokenization method affects sentiment analysis outcomes, providing insights into their strengths and weaknesses in the Korean NLP context. Through this comparative analysis, I hope to inform NLP practitioners on the advantages and limitations of morpheme-based versus subword tokenization methods, with practical implications for Korean-language sentiment analysis and other NLP applications.

# 2 Related Work

## 2.1 Tokenization Approaches in Korean Language Processing

Tokenization is crucial for segmenting text into manageable units, enabling syntactic and semantic analysis in NLP tasks. Korean, characterized by agglutinative structures and complex morphology, presents unique tokenization challenges [? ]. Words often comprise multiple morphemes—units carrying distinct syntactic or semantic information—making it difficult to apply tokenization methods designed for languages with simpler morphological structures.

Traditional Korean tokenizers like MeCab operate at the morpheme level, decomposing words into smaller, meaningful units such as nouns, particles, and verb endings [? ]. This approach effectively captures syntactic nuances, allowing NLP models to accurately parse morphologically complex words. MeCab, being dictionary-based and rule-driven, is frequently employed in tasks like machine translation and dependency parsing, where precise segmentation of linguistic units significantly impacts performance [? ].

Recent subword tokenization techniques, such as Byte-Pair Encoding (BPE) and SentencePiece, adopt a data-driven approach by generating token vocabularies based on statistical frequencies rather than predefined linguistic rules [? ]. SentencePiece, in particular, creates subword units that efficiently manage vocabulary size and mitigate out-of-vocabulary issues. This flexibility makes it well-suited for deep learning applications, where vocabulary control and representation learning are crucial—especially in morphologically rich languages like Korean [? ].

## 2.2 Previous Research in Korean Sentiment Analysis

Sentiment analysis, which involves extracting subjective information and emotional tone from text, is an active research area in Korean NLP. The Naver Sentiment Movie Corpus (NSMC) serves as a primary benchmark dataset, containing thousands of movie reviews annotated with positive or negative sentiments. It enables researchers to explore the nuances of sentiment expressions in Korean [? ].

Studies demonstrate that tokenization quality is critical for sentiment analysis performance, as sentiment-bearing words or morphemes in Korean often convey subtle shifts in tone and intensity [? ]. Korean text, especially in social media and informal reviews, includes emotive particles, abbreviations, and slang that challenge conventional tokenizers. Research highlights how tokenization affects model accuracy by influencing the clarity and interpretability of sentiment expressions [? ]. For instance, morpheme-based tokenizers like MeCab can capture nuances in grammatical endings that signal emotional tone, while subword tokenizers like SentencePiece handle rare words and subwords more flexibly, addressing vocabulary limitations and improving classification performance [? ].

Although comparative studies have examined tokenization methods for Korean language tasks, research specific to sentiment analysis remains limited. Recent findings suggest that morpheme-based tokenizers excel in tasks requiring syntactic precision, such as dependency parsing, whereas subword tokenizers generalize effectively in neural network models where vocabulary size control and representation are critical [? ]. Building on these findings, this paper compares SentencePiece and MeCab in the context of sentiment analysis, providing insights into how each method affects model performance on Korean text.

# 3 Methodology and Experimental Setup

## 3.1 Dataset: Naver Sentiment Movie Corpus (NSMC)

For this study, I utilized the Naver Sentiment Movie Corpus (NSMC), a widely used dataset for sentiment analysis in Korean. The NSMC dataset consists of thousands of Korean movie reviews, each labeled with a binary sentiment (positive or negative). This corpus is well-suited for tokenization experiments due to its informal language style, including slang, abbreviations, and emotive expressions common in online reviews.

**Preprocessing Steps**:

- **Cleaning and Normalization**: Basic cleaning was applied to remove extraneous characters and standardize text.
- **Padding and Truncation**: Sentences were padded or truncated to a fixed length to ensure uniform input sizes, optimizing them for model training.

## 3.2 Tokenization Techniques: SentencePiece and Mecab

Two tokenization methods were applied to the NSMC dataset for comparison: SentencePiece and Mecab.

**SentencePiece**: SentencePiece is a data-driven subword tokenizer that does not rely on predefined linguistic rules. It splits text into subword units based on frequency, resulting in a flexible vocabulary size and fewer out-of-vocabulary words. SentencePiece was configured to generate subwords without removing punctuation, aiming to capture meaningful semantic fragments within words.

**Mecab**: Mecab is a rule-based, morpheme-level tokenizer commonly used in Korean NLP. It segments words into morphemes based on linguistic rules, capturing grammatical nuances critical for syntactic interpretation. For this experiment, Mecab was configured to remove punctuation to focus on semantic elements rather than sentence structure.

### 3.2.1 SentencePiece Tokenizer Setup

The following code snippet shows how the SentencePiece tokenizer was configured and used for subword tokenization.

```
import sentencepiece as spm

# Train SentencePiece model
spm.SentencePieceTrainer.train(input='nsmc.txt', model_prefix='spm',
    vocab_size=32000)

# Load and tokenize using SentencePiece
sp = spm.SentencePieceProcessor(model_file='spm.model')
tokens = sp.encode_as_pieces("영화가 정말재미있었어요 !")
print(tokens)
```

Listing 1: SentencePiece Tokenization Example

**Output:** `[' 영화', '가', ' 정말', ' 재미있', '었', '어요', '!']`

### 3.2.2 Mecab Tokenizer Setup

The following code snippet demonstrates how the Mecab tokenizer was configured for morpheme-based tokenization.

```
from konlpy.tag import Mecab

mecab = Mecab()
tokens = mecab.morphs("영화가 정말재미있었어요 !")
print(tokens)
```

Listing 2: Mecab Tokenization Example

### 3.3 LSTM Model Architecture

The following code snippet outlines the LSTM model architecture used for sentiment analysis.

```
model_lstm = tf.keras.Sequential()
model_lstm.add(tf.keras.layers.Embedding(vocab_size, word_vector_dim,
    input_shape=(None,)))

# First LSTM layer (Bidirectional) - BatchNormalization, Dropout applied
model_lstm.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,
    return_sequences=True)))
model_lstm.add(tf.keras.layers.BatchNormalization())
model_lstm.add(tf.keras.layers.Dropout(0.4))

# Second LSTM layer (Bidirectional) - BatchNormalization, Dropout applied
model_lstm.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)))
model_lstm.add(tf.keras.layers.BatchNormalization())
model_lstm.add(tf.keras.layers.Dropout(0.4))

# Dense layer with L2-normalization
model_lstm.add(tf.keras.layers.Dense(64, activation='relu',
    kernel_regularizer=tf.keras.regularizers.l2(0.02)))
model_lstm.add(tf.keras.layers.Dropout(0.5))

# Output layer
model_lstm.add(tf.keras.layers.Dense(1, activation='sigmoid'))

model_lstm.summary()
```

Listing 3: LSTM Model Architecture

**Output:**

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding  (Embedding)       (None, None, 10)          100000
_________________________________________________________________
bidirectional_1 (Bidirection (None, None, 128)         38400
_________________________________________________________________
batch_normalization_1 (Batch (None, None, 128)         512
_________________________________________________________________
dropout_1 (Dropout)          (None, None, 128)         0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 128)               98816
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
```

```
Total params: 246,561
Trainable params: 246,049
Non-trainable params: 512
-----------------------------------------------------------------
```

### 3.4 Evaluation Setup and Metrics

The sentiment analysis model used for both tokenization approaches was a Bidirectional LSTM (BiLSTM) model, optimized to capture context from both directions in the text.

- **Model Architecture**: The architecture included an embedding layer, two BiLSTM layers, batch normalization, dropout layers for regularization, and dense layers for classification.
- **Hyperparameters**: The BiLSTM model was trained for 20 epochs with a batch size of 1024. Dropout rates were set to 0.4 for the LSTM layer and 0.5 for the dense layer to mitigate overfitting.
- **Evaluation Metrics**: Performance was measured using accuracy, precision, and recall, providing a comprehensive evaluation of each tokenizer's impact.

## 4 Experiments and Results: Tokenization Impact on Sentiment Analysis

### 4.1 Tokenization Analysis

I analyzed each tokenizer's vocabulary size, tokenization speed, and average token length:

- **Vocabulary Size**: SentencePiece created a vocabulary of subwords, which controlled the vocabulary size effectively. Mecab, on the other hand, generated a more extensive vocabulary based on morphemes, capturing syntactic variations.
- **Tokenization Speed**: SentencePiece demonstrated faster tokenization due to its subword-based approach, whereas Mecab, being morpheme-based, took slightly longer.
- **Token Length**: SentencePiece tokens were shorter on average, reflecting its subword structure, while Mecab tokens were longer due to morpheme segmentation, which resulted in fewer tokens per sentence.

### 4.2 Sentiment Model Performance

The table below summarizes the performance of the sentiment analysis model for each tokenization method in terms of accuracy, precision, and recall.

Table 1: Sentiment Model Performance with SentencePiece and Mecab

| Tokenization Method | Criteria for Vocabulary | Accuracy | Precision | Recall |
|---|---|---|---|---|
| SentencePiece | Controlled subword vocabulary | 0.8285 | 0.8078 | 0.8654 |
| Mecab | Extensive morpheme-based vocabulary | 0.8422 | 0.8484 | 0.8355 |

### 4.3 Qualitative Observations

Examples from the tokenized text revealed that SentencePiece captured frequent subwords, which helped in generalizing across vocabulary. However, it sometimes missed sentiment-bearing morphemes. Mecab, with its morpheme-level granularity, was better at capturing sentiment nuances specific to the Korean language, such as mood particles and verb endings, which contribute to the emotional tone of sentences.

# 5 Discussion: Insights on Tokenization for Korean NLP

## 5.1 Interpreting Results in Korean Language Context

The experiments indicate that tokenization method influences sentiment analysis performance. Mecab, with its focus on morphemes, outperformed SentencePiece in this study, likely due to its ability to capture Korean-specific syntactic features that influence sentiment. SentencePiece, while efficient, may lose sentiment nuances by segmenting words into subwords without considering linguistic rules.

## 5.2 Strengths and weaknesses of SentencePiece vs. Mecab

**SentencePiece**: Its data-driven approach makes it versatile and computationally efficient, suitable for applications requiring fast processing and a manageable vocabulary. however, it may lack the linguistic specificity needed for tasks that depend on syntactic or morphological cues, such as sentiment analysis in Korean.

**Mecab**: While slower and more complex, Mecab's rule-based segmentation aligns well with the structural nuances of Korean, making it more effective for sentiment-sensitive tasks. Its reliance on linguistic rules allows it to capture important morphemes, although it may require more computational resources.

## 5.3 Broader Implications for Korean NLP

These findings suggest that morpheme-based tokenization methods like Mecab may be preferable for sentiment analysis and other Korean NLP tasks that depend on syntactic or emotive nuances. however, subword tokenizers like SentencePiece could be more efficient for applications requiring generalization across diverse domains and less dependency on language structure.

# 6 Conclusion and Future Directions

This study compared two tokenization methods—SentencePiece and Mecab—on a sentiment analysis task using the Naver Sentiment Movie Corpus. The results shoId that Mecab's morpheme-based approach was better suited for capturing sentiment in Korean, while SentencePiece offered faster processing and flexibility at the cost of some sentiment-specific detail.

**Future Work**: Further studies could investigate hybrid tokenization methods that combine subword and morpheme-based approaches to balance speed and linguistic precision. Additionally, experiments on other Korean NLP tasks, such as translation or summarization, could offer broader insights into tokenization strategies for Korean text processing.

# References

# A Appendix