



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

언어학박사학위논문

The Construction of a Korean
Pre-Trained Model and an Enhanced
Application on Sentiment Analysis

한국어 사전학습모델 구축과 확장 연구:
감정분석을 중심으로

2021년 2월

서울대학교 대학원
언어학과 언어학전공
이 상 아

The Construction of a Korean Pre-Trained Model and an Enhanced Application on Sentiment Analysis

한국어 사전학습모델 구축과 확장 연구: 감정분석을 중심으로

지도교수 신 효 필

이 논문을 언어학박사 학위논문으로 제출함


2020년 12월

서울대학교 대학원
언어학과 언어학전공
이 상 아

이상아의 박사 학위논문을 인준함

2020년 12월

위원장	남승호	(인)
부위원장	신효필	(인)
위원	강승식	(인)
위원	유현조	(인)
위원	김문형	(인)



Abstract

The Construction of a Korean Pre-Trained Model and an Enhanced Application on Sentiment Analysis

Lee, Sangah

Department of Linguistics

The Graduate School

Seoul National University

Recently, as interest in the Bidirectional Encoder Representations from Transformers (BERT) model has increased, many studies have also been actively conducted in Natural Language Processing based on the model. Such sentence-level contextualized embedding models are generally known to capture and model lexical, syntactic, and semantic information in sentences during training. Therefore, such models, including ELMo, GPT, and BERT, function as a universal model that can impressively perform a wide range of NLP tasks.

This study proposes a monolingual BERT model trained based on Korean texts. The first released BERT model that can handle the Korean language was Google Research's multilingual BERT (M-BERT), which was constructed with training data and a vocabulary composed of 104 languages, including Korean and English, and can handle the text of any language contained in the single model. However, despite the advantages of multilingualism, this model does not fully reflect each language's characteristics, so that its text processing performance in each language is lower

than that of a monolingual model. While mitigating those shortcomings, we built monolingual models using the training data and a vocabulary organized to better capture Korean texts' linguistic knowledge.

Therefore, in this study, a model named KR-BERT was built using training data composed of Korean Wikipedia text and news articles, and was released through GitHub so that it could be used for processing Korean texts. Additionally, we trained a KR-BERT-MEDIUM model based on expanded data by adding comments and legal texts to the training data of KR-BERT. Each model used a list of tokens composed mainly of Hangeul characters as its vocabulary, organized using WordPiece algorithms based on the corresponding training data. These models reported competent performances in various Korean NLP tasks such as Named Entity Recognition, Question Answering, Semantic Textual Similarity, and Sentiment Analysis.

In addition, we added sentiment features to the BERT model to specialize it to better function in sentiment analysis. We constructed a sentiment-combined model including sentiment features, where the features consist of polarity and intensity values assigned to each token in the training data corresponding to that of Korean Sentiment Analysis Corpus (KOSAC). The sentiment features assigned to each token compose polarity and intensity embeddings and are infused to the basic BERT input embeddings. The sentiment-combined model is constructed by training the BERT model with these embeddings.

We trained a model named KR-BERT-KOSAC that contains sentiment features while maintaining the same training data, vocabulary, and model configurations as KR-BERT and distributed it through GitHub. Then we analyzed the effects of using sentiment features in comparison to KR-BERT by observing their performance in language modeling during the training process and sentiment analysis tasks. Addi-

tionally, we determined how much each of the polarity and intensity features contributes to improving the model performance by separately organizing a model that utilizes each of the features, respectively. We obtained some increase in language modeling and sentiment analysis performances by using both the sentiment features, compared to other models with different feature composition. Here, we included the problems of binary positivity classification of movie reviews and hate speech detection on offensive comments as the sentiment analysis tasks.

On the other hand, training these embedding models requires a lot of training time and hardware resources. Therefore, this study proposes a simple model fusing method that requires relatively little time. We trained a smaller-scaled sentiment-combined model consisting of a smaller number of encoder layers and attention heads and smaller hidden sizes for a few steps, combining it with an existing pre-trained BERT model. Since those pre-trained models are expected to function universally to handle various NLP problems based on good language modeling, this combination will allow two models with different advantages to interact and have better text processing capabilities. In this study, experiments on sentiment analysis problems have confirmed that combining the two models is efficient in training time and usage of hardware resources, while it can produce more accurate predictions than single models that do not include sentiment features.

Keywords: Korean embedding model, BERT, language modeling, multi-head self-attention, sentiment features, external fusing of models, sentiment analysis

Student Number: 2016-30038

Contents

1	Introduction	1
1.1	Objectives	3
1.2	Contribution	9
1.3	Dissertation Structure	10
2	Related Work	13
2.1	Language Modeling and the Attention Mechanism	13
2.2	BERT-based Models	16
2.2.1	BERT and Variation Models	16
2.2.2	Korean-Specific BERT Models	19
2.2.3	Task-Specific BERT Models	22
2.3	Sentiment Analysis	24
2.4	Chapter Summary	30
3	BERT Architecture and Evaluations	33
3.1	Bidirectional Encoder Representations from Transformers (BERT)	33
3.1.1	Transformers and the Multi-Head Self-Attention Mechanism	34
3.1.2	Tokenization and Embeddings of BERT	39
3.1.3	Training and Fine-Tuning BERT	42
3.2	Evaluation of BERT	47
3.2.1	NLP Tasks	47

3.2.2	Metrics	50
3.3	Chapter Summary	52
4	Pre-Training of Korean BERT-based Model	55
4.1	The Need for a Korean Monolingual Model	55
4.2	Pre-training Korean-specific BERT Model	58
4.3	Chapter Summary	70
5	Performances of Korean-Specific BERT Models	71
5.1	Task Datasets	71
5.1.1	Named Entity Recognition	71
5.1.2	Question Answering	73
5.1.3	Natural Language Inference	74
5.1.4	Semantic Textual Similarity	78
5.1.5	Sentiment Analysis	80
5.2	Experiments	81
5.2.1	Experiment Details	81
5.2.2	Task Results	83
5.3	Chapter Summary	89
6	An Extended Study to Sentiment Analysis	91
6.1	Sentiment Features	91
6.1.1	Sources of Sentiment Features	91
6.1.2	Assigning Prior Sentiment Values	94
6.2	Composition of Sentiment Embeddings	103

6.3	Training the Sentiment-Combined Model	109
6.4	Effect of Sentiment Features	113
6.5	Chapter Summary	121
7	Combining Two BERT Models	123
7.1	External Fusing Method	123
7.2	Experiments and Results	130
7.3	Chapter Summary	135
8	Conclusion	137
8.1	Summary of Contribution and Results	138
8.1.1	Construction of Korean Pre-trained BERT Models	138
8.1.2	Construction of a Sentiment-Combined Model	138
8.1.3	External Fusing of Two Pre-Trained Models to Gain Performance and Cost Advantages	139
8.2	Future Directions and Open Problems	140
8.2.1	More Training of KR-BERT-MEDIUM for Convergence of Performance	140
8.2.2	Observation of Changes Depending on the Domain of Training Data	141
8.2.3	Overlap of Sentiment Features with Linguistic Knowl- edge that BERT Learns	142
8.2.4	The Specific Process of Sentiment Features Helping the Language Modeling of BERT is Unknown	143

A.	Python Sources	157
A.1	Construction of Polarity and Intensity Embeddings . . .	157
A.2	External Fusing of Different Pre-Trained Models	158
B.	Examples of Experiment Outputs	162
C.	Model Releases through GitHub	165

List of Figures

1	The Process of Training and Fine-tuning BERT	33
2	The model architecture of the Transformer (Vaswani et al. 2017)	35
3	Attention calculation (Vaswani et al. 2017)	36
4	BERT embeddings (Devlin et al. 2019)	41
5	Visualization of Masked Language Modeling	43
6	Visualization of Next Sentence Prediction	44
7	Training loss of KR-BERT	67
8	Training loss of KR-BERT-MEDIUM	68
9	Comparison of sentiment word matching depending on the lexicon	98
10	Visualization of the embedding composition	105
11	Training loss of KR-BERT-KOSAC	110
12	Attention maps for Example (9)	119
13	Attention map for Example (10)	120
14	The structure of our external fusing method	123
15	Training loss of KR-BERT-KOSAC-small	127
16	Training loss of CH-BERT-KOSAC-small	129

List of Tables

1	Confusion Matrix	51
2	Comparison of MLM performance for KR-BERT according to vocabulary size	60
3	Comparison of MLM performance for KR-BERT-MEDIUM according to vocabulary size	62
4	Composition of the vocabularies	62
5	Details of Korean pre-trained BERT models	64
6	Training performances of KR-BERT and KR-BERT-MEDIUM	68
7	The category of the named entities	72
8	Rule of thumb for interpreting the size of a correlation coeffi- cient (Hinkle et al. 2003)	79
9	The model performances for Korean NLP tasks	83
10	The distribution of the polarity tags in KOSAC	93
11	The distribution of the intensity tags in KOSAC	93
12	The distribution of the polarity tags in KNU Sentiment Lexicon	94
13	The alignment of sentiment values in KOSAC and the KNU lexicon	97
14	The ratio of sentiment words in the KR-BERT vocabulary . . .	99
15	The ratio of sentiment words in the KR-BERT training data . .	100

16	Training performances of models with different sentiment word compositions	102
17	An example of tokens and assigned sentiment values	104
18	Training performances of the models predicting and masking tokens and sentiment values	108
19	Training performances of KR-BERT and KR-BERT-KOSAC	110
20	Model performances on sentiment analysis tasks	112
21	Training performances of ablated models	114
22	Task performances of ablated models	115
23	Training performances of KR-BERT-KOSAC-small	126
24	Training performances of CH-BERT-KOSAC-small	128
25	Task performances of externally-fused models	131

1 Introduction

Recently, in many areas, people attempt to obtain information by processing massive text data. Governments get opinions from online forums on how people feel about policies, and companies obtain reviews from consumers about their products from websites. Processing news articles can also help analyze social phenomena. These datasets are too large for a human to process manually, so such text processing is left to machines. It is with this type of work that Natural Language Processing (NLP) allows people to obtain and analyze the information they want through automatic processing from texts.

In NLP, a specific domain or purpose model is trained and produces the desired outputs for various tasks. Then the outputs are evaluated by how accurately they were produced, which becomes the tasks' performances. For example, if a model executes a binary classification problem about product reviews that predicts whether each review's stance is positive or negative, the model first converts the review texts in the training dataset into a proper vector form that the model architecture can handle. The vector is organized by various methods, from word frequencies to sentence embeddings. The model is trained based on these vectors and then predicts the label on the reviews in the test dataset using a classifier layer, usually located on the last stage of the model architecture. The classifier calculates the probabilities in which each review has a positive or negative stance and selects the one with a higher probability as the model prediction.

These machine learning processes obtain remarkable performances using various word embedding models and linguistic knowledge with neural network models. However, they require a separate dataset for training and evaluation for each task and domain, and the datasets should be annotated with labels for prediction. Moreover, it is not easy to get enough data for all tasks. Some tasks do not have data published for them, or the data available are very insufficient and may not have sufficient quality. Data collection and labeling is expensive for researchers to build on their own since they take a lot of labor and time. In particular, this problem is more serious when it comes to low resource languages other than English.

Pre-trained contextualized embedding models, such as ELMo (Peters et al. 2018), GPT (Radford et al. 2018), and ULMFiT (Howard and Ruder 2018), have been released to mitigate such shortcomings. They are trained based on large unlabeled texts, having the general purpose of being applied to various NLP tasks. The texts used for model training are not for a specific task, so they do not need annotations of task labels. Instead, the model learns general linguistic knowledge from the training data on its own through language modeling, and produces the output language representations, that is, embeddings that include the linguistic features it has learned. Thus, the embeddings constructed from the model contain much more complex and various linguistic phenomena than simple vectors using hand-crafted features such as word frequencies. Such linguistic phenomena range from lexical information to semantic, syntactic, and pragmatic information, which require a broader context

span to process.

The constructed embeddings can be applied to various NLP tasks, with task-specific classifier layers added on top of the embeddings' final hidden states. For each task, the classifier fine-tunes the pre-trained embedding model according to the task, and the task datasets for fine-tuning do not need to be too large. Moreover, the general embedding model has good performance in each task since it contains universal linguistic knowledge.

Amid this trend, Vaswani et al. (2017) released a Transformer architecture using a multi-head self-attention mechanism, and Google Research provided a contextualized embedding model, BERT (Devlin et al. 2019), based on the encoders of Transformer. BERT is also a pre-trained language representation model like previous models, but it has become a significant turning point in NLP research by showing state-of-the-art performance in various tasks in comparison to previous models. Accordingly, many studies have begun to provide BERT-derived works, such as applying BERT to various tasks or domains, building models for each language other than English, and creating models that improve BERT's internal architecture.

1.1 Objectives

The Bidirectional Encoder Representations from Transformers model, BERT (Devlin et al. 2019), is a contextualized embedding model originally trained based on English texts and applied to English NLP tasks. It reported much higher performances than other existing models in several English NLP tasks,

but could not be applied to tasks based on other languages. Following this, however, Google Research released the Multilingual BERT (M-BERT) model, which is trained based on texts of 104 different languages, including Korean, English, German, and Finnish. It has the advantage of being able to handle multiple languages, but it is insufficient for handling all monolingual tasks well.

(1) 최고의 홍콩 액션영화였다.

choykouy hongkhong ayksyenyenghwayessta.

“It was the best Hong Kong action movie.”

Example (1) is an excerpt from a Korean Movie Review Dataset¹. M-BERT tokenizes it as below.

최고 ##의 흥 ##콩 액 ##션 ##영 ##화 ##였다 .

choyko ##uy hong ##khong ayk ##syen ##yeng ##hwa ##yessta .

In this case, the word 홍콩 hongkhong “Hong Kong” is split into ‘흥 hong’ and ‘##콩 ##khong’, and 액션영화 ayksyenyenghwa “action movie” is split into ‘액 ayek,’ ‘##션 ##syen,’ ‘##영 ##yeng,’ and ‘##화 ##hwa.’ The BERT model splits words into subword units to minimize out-of-vocabulary (OOV) words. On the other hand, using a model that is trained based on Korean texts only, the same example is tokenized as below. The tokenization is executed

¹<https://github.com/e9t/nsmc>

by the KR-BERT model constructed in this dissertation.

최고의 홍콩 액션 ##영화 ##였다.

choykouy hongkhong ayksyen ##yenghwa ##yessta .

Here, 홍콩 hongkhong “Hong Kong” is not split, maintaining its original form, and 액션영화 ayksyenyenghwa is split into ‘액션 ayksyen “action” and ‘##영화 ##yenghwa “movie”.’ Each tokenized unit maintains its meaning and will function better in combining with surrounding words and modeling the whole context of a sentence than splitting into subwords such as ‘홍 hong, ##콩 ##khong, 액 ayk, ##션 ##syen, ##영 ##yeng, ##화 ##hwa,’ which do not indicate correct meanings themselves.

Therefore, to address these cases and mitigate the shortcomings of the multilingual model, several studies have constructed monolingual models in different non-English languages (Antoun et al. 2020; Cañete et al. 2020; Hussein 2018; Kikuta 2019; Kłeczek 2020; Nguyen and Nguyen 2020; Polignano et al. 2019; Virtanen et al. 2019; Vries et al. 2020). The same research was also done in Korean, releasing several pre-trained models: ETRI KorBERT², SKT KoBERT³, and TwoBlockAI HanBERT⁴.

Such models have been known to learn linguistic knowledge independently, but several studies use additional linguistic features (Huang, Sun, et

²http://aiopen.etri.re.kr/service_dataset.php

³<https://github.com/SKTBrain/KoBERT>

⁴<https://github.com/tbai2019/HanBert-54k-N>

al. 2019; Levine et al. 2020; Zhang et al. 2019; Zhou et al. 2020). NLP tasks associated with the features are improved by applying additional information to BERT-based models in various ways. There are also such studies in sentiment analysis (Ke et al. 2020; Tian et al. 2020).

For sentiment analysis, sentiment features are used, in many cases constructed using words including polarity or sentiment words. While there are many branches of studies such as aspect and emotions for sentiment analysis, we focus on classifying the binary stance of sentences or documents to positive or negative. For instance, Example (2) is a film review, and this review has a positive stance for the film. Here, the reviewer uses expressions such as 웃기고 wuskiko “funny” and 재밌었다 caymissessta “interesting” to give a positive review of the film, and the reader can understand that the review is optimistic about the film by referring to these words.

(2) 이 영화 너무 웃기고 재밌었다.

i yenghwa nemwu wuskiko caymissessta.

“This movie was so funny and interesting.”

Polarity Label: Positive

In other words, information related to sentiment, such as sentiment words, will help the stance judgment of sentences, and this will also be applied to the training, fine-tuning, and prediction of the model. Therefore, it is essential to select and process additional linguistic information, such as sentiments, properly.

Based on these phenomena as background, this study aims to build a Korean BERT model so that it functions well in overall NLP tasks, release the model publicly, and improve the performance of sentiment analysis tasks by applying additional information or features and additional methodologies to the model.

In pursuit of this goal, we seek to establish and confirm the following hypotheses.

1. The Korean monolingual model will better reflect the Korean language's characteristics and handle Korean texts better than the multilingual model.
2. Further construction of an expanded Korean model using more extensive and more diverse training data will help process Korean texts and related NLP tasks.
3. Infusing additional sentiment features to these BERT-based Korean models will help understand the meaning of the sentences in the datasets of sentiment analysis by capturing and processing the sentiment information required for the tasks.
4. The sentiment features will be better at processing the semantic and sentiment information of texts when configured using both polarity and intensity information of words than when using only one.
5. A new methodology that combines the pre-trained model with a newly-trained model using sentiment features will help processing sentiment analysis through interactions between the two models.

The hypotheses above will be tested in this dissertation indirectly through experimental results. First, the first hypothesis will be tested by applying the Korean-specific BERT model trained in this study to several Korean NLP tasks from various domains and comparing them with other existing Korean models' performances. The tasks consist of Named Entity Recognition (NER), Question Answering (QA), and the detection of similarity or entailment between sentences and sentiment analysis.

We examine the second hypothesis by training an expanded model by adding data from different sources to the Korean language model's training data and comparing its performances on Korean NLP tasks with those of the original model with smaller training data.

The sentiment features are composed and added to the input embeddings of the Korean-specific BERT models trained in this study to test the third hypothesis. In training the sentiment-combined model, the effects of the sentiment features are identified through Masked Language Modeling (MLM) performance, one of BERT's training subtasks. The model trained with additional sentiment features is also applied to sentiment analysis tasks and evaluated by its performance.

The fourth hypothesis will be examined by an ablation study in which the model performance is obtained by subtracting each of the polarity and intensity features that compose our sentiment features from the sentiment-combined model. Here, the model's training performance, especially MLM accuracy and loss, and the performance in sentiment analysis tasks are evalu-

ated.

Finally, we test the fifth hypothesis by combining a pre-trained, general purpose language model and a sentiment-combined model using sentiment features to apply to sentiment analysis, and compare the performance with those of the existing single pre-trained models.

1.2 Contribution

We constructed the BERT-based models using Korean training data, which can be applied to several Korean NLP tasks. Moreover, we released the trained models through GitHub to make them easy to access and utilize when processing Korean texts. There are not many task data released in Korean, and fewer are both sufficient in quantity and released publicly. Therefore, providing pre-trained, generic purpose models will be helpful.

Such Korean pre-trained models will also contribute to the study of theoretical linguistics, Korean linguistics, and natural language processing. In academic fields, more and more computational methods are used, and recently, texts are analyzed using word embedding models such as Word2vec (Mikolov et al. 2013) or GloVe (Pennington et al. 2014). However, using sentence-level embedding models such as BERT would help process the meaning of texts and linguistic phenomena, using context in a broader range than words and n-grams, such as sentences or documents.

Additionally, we propose a new method of using a specific set of new features for sentiment-related tasks, and release a model which uses this method.

We constructed sentiment features using polarity and intensity information from a Korean sentiment lexicon and trained a sentiment-combined model using them. We also provide a methodology for externally fusing the Korean pre-trained language models and sentiment-combined models to apply to sentiment analysis tasks. As a result of adding sentiment features and combining the trained models in a new way, we could improve Korean sentiment analysis performance.

Furthermore, our novel external sentiment fusing method saves training time and effort. The basic pre-trained general purpose models are conveniently imported without additional training, and the newly-trained sentiment-combined models are trained using a small-scale architecture, smaller training steps, and low level hardware. After that, only fine-tuning is needed, and is done by applying both models to the sentiment-related tasks at the same time. Doing so shows improved performance over when the models are used alone.

1.3 Dissertation Structure

This chapter presents the research problems of constructing Korean pre-trained, general purpose, contextualized embedding models for several NLP tasks, and training sentiment-combined models using related features to improve performance on sentiment analysis tasks.

Chapter 2 introduces previous studies, including BERT and the following BERT-based models improved to construct more effective or efficient systems. Studies on sentiment analysis are also mentioned, ranging from tradi-

tional models using hand-crafted features to neural networks and BERT-based models.

To explain our approach using BERT, we describe the basic architecture of the Transformer and BERT in Chapter 3. The process of training, fine-tuning, and testing BERT models is explained, with popular NLP tasks and standard evaluation metrics given for them.

Chapter 4 describes the Korean-specific monolingual BERT-based models trained in this study, with their training data, vocabulary, and parameters. These training details are compared with those of other existing Korean models.

Chapter 5 evaluates the models introduced and trained in Chapter 4 by applying them to various Korean NLP tasks and measuring their performance against other Korean pre-trained models.

We introduce sentiment features and infuse them to the language models above to better process the sentiment analysis tasks in Chapter 6. The sentiment-combined model’s training and task performance is compared to those of the original models without sentiment features. Moreover, ablation studies are executed to observe the effects of using the sentiment features and polarity and intensity information.

Chapter 7 describes a different, new methodology of using sentiment features to improve BERT-based models’ performance in sentiment analysis more clearly. We combine two different purpose models, one for general purpose use and the other for sentiment analysis, and compare the models’ perfor-

mances to verify the effect of the external fusing of different models.

Finally, Chapter 8 summarizes this dissertation's main findings and discusses the remaining problems with future directions to improve and expand our study.

2 Related Work

This chapter introduces previous work related to the content of this dissertation. First, 2.1 describes language models and previous studies on them. Additionally, we mention the attention mechanism applied to models for language modeling. 2.2 refers to the BERT model on which the models provided by this study are based. We also introduce the various derivations of BERT, including Korean-specific and task-specific pre-trained models. In 2.3, we describe previous studies on sentiment analysis that we address, expanding upon our Korean-specific BERT models.

2.1 Language Modeling and the Attention Mechanism

A Language Model is a probabilistic model that calculates the probability of the occurrence of a token in a sequence. The probability is calculated by predicting a word given the previous words in a sequence, based on conditional probabilities, as in Equation (1). The equation computes the probability of a sequence W consisting of words w_1, w_2, \dots , calculating the occurrence probability of the n^{th} word given the $n - 1$ previous words in the sequence.

$$P(W) = P(w_1, w_2, w_3, \dots, w_n) = \prod_{i=1}^n P(w_i | w_1, \dots, w_{i-1}) \quad (1)$$

Here, the sequence with a higher probability is judged to be more proper. This method can be applied to several fields, including machine translation, spelling correction, and speech recognition. Traditionally, the n-gram lan-

guage modeling method was utilized to limit the number of previous words used for a word prediction. Its use was limited to counting and modeling phrases in a training corpus, as it would be hard for such n-gram models to count longer sequences. Since such language modeling still could not learn about sentences not included in the training data, neural language models and word embedding models attempted to model such sentences by training for the similarities between words (Yoo 2020).

These neural language models project the words' presence and frequency information in the form of embedding vectors. Models such as RNNs, LSTMs, and GRUs execute language modeling based on vectors calculated through multiple hidden layers. The embedding vectors include relations between words, including word similarity, since they represent words in more detail with many figures. Based on this idea, various word embedding models such as Word2Vec (Mikolov et al. 2013), GloVe (Pennington et al. 2014), and Fast-Text (Bojanowski et al. 2017) were widely used.

Meanwhile, development into contextualized representation models was beginning. Such contextualized models learn text representations via language modeling, mainly based on an AutoRegressive (AR) method, which predicts a token given the previous tokens in a sequence, as in several of the neural language models above. Through the AR method, the likelihood of an input sequence is calculated by multiplying the conditional probabilities of the n-grams in a sequence.

ELMo (Peters et al. 2018) and GPT (Radford et al. 2018) are trained via

the AR method. ELMo obtains the final hidden states of multi-layer LSTMs, which consider both directions of a sequence, forward and backward, and concatenates them to learn bidirectional context information in text sequences. GPT utilizes a modified version of the decoder of the Transformer architecture. It trains the language model via unsupervised language modeling based on the AR method's objective function, and fine-tunes the model using task-specific labeled datasets.

On the other hand, some models use an Auto Encoding (AE) method for language modeling. A model, given an input sequence, predicts the same sequence to learn text representations. Through the AE method, the language model obtains the bidirectional contextual information of a sequence without additional concatenation, unlike AR models. BERT (Devlin et al. 2019) is a representative AE model that predicts the original sequence given a corrupted sequence with masked tokens (Denoising Auto Encoder). BERT is constructed based on the Transformer (Vaswani et al. 2017), specifically its multi-head self-attention mechanism.

The attention mechanism, the basis of the Transformer architecture and BERT, is a mechanism used to improve neural network models' performances. Bahdanau et al. (2015) first provided the method to search for words in a source sentence relevant to and to be focused on for predicting a target word, to improve machine translation performance.

Several studies added the attention mechanism to language models to help them learn linguistic knowledge from texts. Sukhbaatar et al. (2015) proposed

an end-to-end memory network based on an RNN, but included layer-wise weights and divided the RNN output into internal and external outputs. The internal output considers memory, which is analogous to attention weights, and the external output corresponds to predicting a label. Daniluk et al. (2017) applied a key-value attention mechanism, in which the Key and Value parts are separately considered from the output vector to by neural language models. Salton et al. (2017) provided an Attentive RNN-LM that applies the attention mechanism to its inputs, encoded by a multi-layered RNN model.

The attention mechanism has also been applied to improve performance in various NLP tasks, including: machine translation, text classification, text summarization, information extraction, semantic tasks including textual entailment and semantic role labeling, syntactic parsing, and sentiment analysis (Galassi et al. 2020). The detailed processes of computing attention for a sequence are explained in detail later in 3.1.1.

2.2 BERT-based Models

2.2.1 BERT and Variation Models

BERT is a contextualized embedding model, which is a refined and improved version of the encoder part of the Transformer. The model is trained based on a sizeable unlabeled corpus through bidirectional language modeling to construct language representations that can be applied to several tasks, providing robust performance in broad fields of NLP. Its training process uses two internal subtasks: Masked Language Modeling (MLM), in which the relation

between the tokens in a sentence is learned; and Next Sentence Prediction (NSP), where the relation between sentences in a document are modeled.

This BERT model was trained based on English texts, and most of the BERT-based models described and improved here are also English-based. Multilingual models that handle multiple languages at a time have also been released. Moreover, some models have been trained using a specific non-English language corpus.

Since then, various BERT variation models have appeared. First of all, there is a trend to construct huge models that use a larger scale architecture and more parameters than BERT to achieve improved performance. XLNet (Yang et al. 2019) used additional text to BERT's training data and considered permuted sequences to mask and predict all tokens in the text, whereas BERT masks and predicts only 15% of tokens, accessing the tokens in sentences in sequential order. This makes the model better at learning the dependencies and relations between words and achieve higher performance than BERT. However, higher hardware specifications are required due to its large number of calculations.

RoBERTa (Liu et al. 2019), released by Facebook, also used additional training data and used a dynamic masking method that removed NSP and improved MLM from BERT's original training tasks. It does this by masking different tokens according to its training epoch. It also reported higher performance than BERT in many tasks while requiring higher hardware specifications due to its high number of computations.

There have also been smaller models developed to try and overcome the limitations of hardware or the long training times that such huge models require. DistilBERT (Sanh et al. 2019) is a model that removes token-type embedding and poolers from the BERT model and halves the number of parameters calculated by halving the number of layers. It is reported that DistilBERT’s performance remained at 97% of BERT’s performance even though its computation has been reduced.

ALBERT (Lan et al. 2020) reduced the model’s size by applying two parameter reduction techniques to the BERT model: factorized embedding parameterization that reduces the number of input layers’ parameters, and cross-layer parameter sharing. This model suggested Sentence-Order Prediction (SOP) instead of BERT’s NSP task, pointing out that NSP was too easy, and that it is therefore ineffective as a training task. In SOP, the model judges a swapped pair of two consecutive sentences as a false example, and this becomes a task aimed at capturing the discourse-level coherence of a text.

ELECTRA (Clark, Luong, et al. 2020) performed MLM using a separate small generator network through a Generative Adversary Network (GAN) style methodology, and performed more efficient MLM by determining whether each token is a corrupted token by using a discriminative model. Here, the size of the generator model with token corruptions can be reduced, so the number of calculations is also reduced.

There are also various other models, and most of those models are released in Facebook’s Transformers library (Wolf et al. 2020), in the form of a pack-

age for easy use. In this dissertation, modified Korean-specific models were trained based on BERT, which is the basis of the various models above.

2.2.2 Korean-Specific BERT Models

The first BERT model for dealing with Korean texts was the multilingual BERT (M-BERT), which was pre-trained by Google Research using training corpora of 104 languages and a vocabulary constructed from those languages. The model applies to tasks for all 104 language. The languages include Korean, English, German, Arabic, Slovenian, and Turkish, which are chosen as the languages with the most massive Wikipedia data.

This model is easy to import and produces satisfactory performance across its languages, but has many disadvantages compared to subsequent monolingual models trained on individual languages' corpora. The monolingual models capture the characteristics of a language well and perform better in downstream tasks. Starting with BERT, the actual English-only model, monolingual models based on German¹, Italian (Polignano et al. 2019), Finnish (Virtanen et al. 2019), Japanese (Kikuta 2019), Spanish (Cañete et al. 2020), Malaysian (Husein 2018), Dutch (Vries et al. 2020), Arabic (Antoun et al. 2020), Vietnamese (Nguyen and Nguyen 2020), and Polish (Kłeczek 2020) reported higher performances than multilingual models in various NLP tasks.

Korean-specific models also reported higher performances than M-BERT. ETRI released KorBERT, which includes morpheme-level and character-level

¹<https://deepset.ai/german-bert>

models trained using large-scaled training data (23GB) and vocabularies (30,349 morphemes and 30,797 character tokens), and many parameters (110M). The morpheme-level model used an external part-of-speech tagger to construct and model the vocabulary using segmented morpheme units as a basic token. The character-level model tokenized its training data using a similar method to the WordPiece model, a BERT tokenizer model that combines syllable characters that frequently occur in words.

SKT Brain’s KoBERT is a character-level model that constructed its vocabulary and tokenized texts using the SentencePiece tokenizer (Kudo and Richardson, 2018). The SentencePiece tokenizer uses the Unigram Language Model, which examines all possible fractional units in a text and chooses the unit with the highest probability among them as a token in the vocabulary. The training data of KoBERT consist of Korean Wikipedia (5M sentences, 54M words) and news data (20M sentences, 270M words), the vocabulary consists of 8,002 tokens, and the model has 92M parameters.

HanBERT, released by TwoBlock AI, is a morpheme-level model using Moran, a self-developed part-of-speech tagger. Moran performs a morphological analysis using a database composed of about 3 million words and 445,701 words from Wikipedia entries. The training data consists of generic documents (3 billion sentences, 11.3 billion morphemes, 70GB), and patent documents (3.7 billion sentences, 15 billion morphemes, 75GB). The vocabulary includes 54,000 tokens and is based on the entries of the Moran database.

We also trained Korean-specific monolingual BERT models in this dis-

sertation. First, KR-BERT is a character-level model, and the training data consists of Korean Wikipedia texts and news articles (20M sentences, 233M words, 2.47GB). Its vocabulary includes the tokens trained using the WordPiece tokenizer of BERT, and some foreign language characters and special tokens included to deal with casual writings, to form a list of 16,424 tokens. The model contains 99M parameters. KR-BERT is the basis of the sentiment-combined models, which will be explained later in this dissertation.

Additionally, based on KR-BERT, we trained an additional KR-BERT-MEDIUM model with training data expanded in terms of both scale and the variety of domains covered. We will explain it in detail in Chapter 4.

KorBERT and HanBERT use large-scale training data and vocabularies, and a large number of parameters, requiring high memory usage and extended training time. Furthermore, HanBERT and the morpheme-based model of KorBERT set the basic token unit as a morpheme, and thus it is necessary first to preprocess all texts used for training and testing the models with external part-of-speech taggers. This dramatically increases the time, effort, and cost of text processing. While KoBERT uses a smaller vocabulary and a smaller number of parameters than KR-BERT for training, its training data size is slightly larger than that of KR-BERT. Despite the relatively small-scale configurations of KR-BERT, the model reported similar or higher performance than others in various NLP tasks, such as Movie Review Classification, Named Entity Recognition, and Question Answering.

2.2.3 Task-Specific BERT Models

Applying the BERT model to various NLP tasks, there are several additional improvements to model fine-tuning for each task. First of all, LIMIT-BERT (Zhou et al. 2020) infused BERT with syntactic and semantic information, and modified it to be applicable to multiple linguistic tasks: part-of-speech tagging, constituent parsing, dependency parsing, and span- and dependency-level Semantic Role Labeling.

ERNIE (Zhang et al. 2019) is a general-purpose BERT model using entity- and phrase-level knowledge masking strategies. The model reported slightly better performance than other models in five major Chinese NLP tasks.

GlossBERT (Huang, Sun, et al. 2019) used gloss knowledge, the sense definition of words, to perform Word Sense Disambiguation (WSD). They constructed context-gloss pairs from glosses of all possible senses of words in WordNet (Fellbaum 1998), and defined the WSD task as a sentence-pair classification problem.

SenseBERT (Levine et al. 2020) is also an improved BERT-based model for WSD. The model added Masked Word Sense Prediction to the training tasks of BERT to predict masked tokens and their WordNet supersenses simultaneously. By using word-sense information in the pre-training of BERT, the model’s lexical understanding is strengthened.

Moreover, some studies have trained models with corpora from certain domains to make them function well in those fields. BioBERT (Lee, Yoon, et al. 2019) was initialized with the original BERT model and further trained

using biomedical texts (PubMed abstracts and PubMed Central full-text articles). This model was fine-tuned and performed in tasks such as Named Entity Recognition, Relation Extraction between named entities, Question Answering, showing better performance than previous approaches to the tasks.

SciBERT (Beltagy et al. 2019) is a model trained based on scientific publications (Semantic Scholar) and a vocabulary constructed using the SentencePiece algorithm. The model was applied to NLP tasks such as Named Entity Recognition, PICO Extraction (a specific entity recognition on clinical trial paper), Text Classification, Relation Classification (between named entities), and Dependency Parsings, for texts of the biomedical domain and computer science domain. It recorded higher performance than BERT on these tasks.

Huang, Altosaar, et al. (2019) and Alsentzer et al. (2019) constructed domain-specific BERT models trained based on clinical notes. ClinicalBERT (Huang, Altosaar, et al. 2019) trained a BERT-based model with medical notes (MIMIC-III Database) and fine-tuned the model with Hospital Readmission Prediction, which is a clinical task determining the readmission of a patient within the next 30 days. The model reported higher performance than the baselines, bag-of-words, and BiLSTM models.

Alsentzer et al. (2019) initialized their BERT-based models from the existing BERT and BioBERT models, and then fine-tuned them based on the MIMIC-III Database. The models were evaluated on a MedNLI language inference task and four named entity recognition tasks in the clinical domain, recording higher performance with the clinically fine-tuned models than the

non-clinical basic models in many cases.

2.3 Sentiment Analysis

We performed the Sentiment Analysis tasks here by constructing expanded Korean-specific BERT-based models with additional sentiment features. Therefore, we describe the necessary information and previous studies on sentiment analysis here.

Sentiment Analysis is a field of NLP studies in which a model finds the opinion, stance, or feelings in a text that its writer possesses on a particular subject or target. The model is trained based on the text and predicts sentiment labels for it. Texts subject to sentiment analysis tasks span various domains, including product or movie reviews and web texts generated by users, including texts posted on social network services such as Twitter, Reddit, blogs, and discussion forums. Sentiment analysis tasks also vary depending on domain data and the problem definition, such as phrase-level, sentence-level, document-level, and aspect-level sentiment classifications. The granularity of sentiment information also categorizes them into subjectivity classification, polarity classification, and multi-class sentiment type classification.

Several models have been designed and use features that capture sentiment information well. Using proper linguistic features to help with the task is a trend that has been developing for a long time. The features widely used include the existence and frequency of words (bag-of-words), certain words in the text (Benamara et al. 2007; Turney 2002; Wilson et al. 2005), part-of-

speech information, parse tree of expressions, and stylistic properties (Rahate and Emmanuel 2013).

Many studies have used a pre-defined sentiment lexicon as sentiment features. A sentiment lexicon is a kind of dictionary, including sentiment orientation such as the polarity or intensity of words. Sentiment lexicons constructed in English, the most studied language, include SentiWordNet (Baccianella et al. 2010), MPQA (Wiebe et al. 2005), and the Bing Liu Opinion Lexicon (Hu and Liu 2004). SentiWordNet contains the degrees of positivity, negativity, and neutrality (objectiveness) of words annotated for all the WordNet synsets. MPQA and Bing Liu Opinion Lexicon defined a list of words with positive and negative polarity, and the former also annotated the intensity of word subjectivity.

Released Korean sentiment lexicons include the Korean Sentiment Analysis Corpus, KOSAC (Shin et al. 2012), and the KNU Korean Sentiment Lexicon². KOSAC annotated 17,582 sentiment expressions from the Sejong Corpus and news articles. The entries are based on morpheme units, with types of subjectivity, polarity, intensity, and manner of expressions, and their probability values annotated to each of them. Among these features, we used the polarity and intensity values to construct the sentiment features.

The KNU Korean Sentiment Lexicon, released by Kunsan National University, includes 14,843 sentiment words and phrases. The lexicon consists of positive and negative words from several sources: the results of BiLSTM

²<https://github.com/park1200656/KnuSentiLex>

classification on the glosses of the Standard Korean Language Dictionary of the National Institute of the Korean Language, the translated word lists of SentiWordNet and SenticNet-5.0, the word list provided by Gim (2004), and recently popular online abbreviations and emoticons.

Such sentiment values of tokens in a sentence help to predict the sentence-level sentiment label, as in Example (3), which is an excerpt from a movie review in the Naver Sentiment Movie Corpus. In Example (3), the tokens 유쾌 yukhway “pleasant” and 이쁜 ippun “pretty” have an apparent positive stance on the movie, and 너무 nemwu “very” serves as an intensifier to amplify the positivity. Therefore, the sentiment values help us to judge this review as positive.

- (3) 너무 유쾌하고 이쁜 영화
nemwu yukhwayhako ippun yenghwa
“a very pleasant and pretty movie.”
Polarity Label: Positive

Traditional studies using features based on sentiment lexicons are as follows. Teng et al. (2016) built a lexicon-based weighted sum model using prior sentiment scores from sentiment lexicon features as weights in an LSTM. The model produced sentence-level sentiment labels.

Qian et al. (2017) modeled the linguistic role of sentiment, negation, and intensity words to construct the training objectives of an LSTM using linguistic regularizers. The linguistic regularizers allowed the model to predict

each position's sentiment distribution in a sequence according to sentiment, negation, and intensity words.

Sun et al. (2018) encoded the words from a sentiment lexicon, along with the dependency and arguments (personal beliefs) in texts, through an LSTM, and calculated each word's attentions to act as the weights. They obtained semantic vectors by computing a weighted sum of the hidden states of words.

Bao et al. (2019) utilized lexicon information to perform aspect-based sentiment analysis. They defined five categories of sentiment values to each word and used them as the features used in LSTM.

Moreover, Li and Caragea (2019) used the sentiment lexicon of Hu and Liu (2004) and the self-built stance lexicon to encode whether each word in a sequence is included in either of the lexicons, and vectorized the information.

Some works trained embedding models that help the sentiment encoding of a text. Tang et al. (2014) provided an extensive version of the embedding model (Collobert et al. 2011). Another similar model of Fu et al. (2018) expanded the Continuous Bag-of-Words (CBOW) model (Mikolov et al. 2013). These models used a corrupted embedding method, which masks random words in a given text span, making the model learn syntactic and semantic information through language modeling and sentiment information through annotated polarity labels of the training corpus.

On the other hand, it is known that contextualized embedding models such as ELMo and BERT encode linguistic information themselves (Pennington et al. 2014). Such a model can automatically and internally learn linguistic

features, including sentiment features, to produce the same or somewhat better performance than existing models, without hand-crafting features. Therefore, many studies attempt to improve embedding models mathematically, without adding features, as in 2.2.1. It is reported that such models learn sentiment information on their own and produce high performance in sentiment analysis tasks.

However, several studies added features to BERT models and reported improved performances in the related tasks. In particular, BERT-based studies have used additional sentiment information to help to perform sentiment analysis. SentiLARE (Ke et al. 2020) obtained sentiment polarity information from the SentiWordNet dictionary using words and their part-of-speech tags, and trained new BERT models using word sentiment polarity and part-of-speech information with pre-defined sentence-level sentiment labels in the training data. The models were trained based on a review dataset including sentiment labels (Yelp Dataset Challenge 2019).

They modified BERT’s MLM training task to predict part-of-speech tags and the sentiment polarity labels of tokens, as well as the masked tokens themselves, summing the losses of all these predictions. This allows the model to capture the implicit relations among sentence-level labels, tokens, and token information. The models were tested on English sentence-level sentiment classification datasets and aspect-based sentiment analysis tasks, recording higher performance than existing models.

SKEP (Tian et al. 2020) is a model that included sentiment words and

aspect-sentiment pairs in the model training process. It was based on RoBERTa since it did not use the NSP task during training. The authors annotated the polarity of words in a sequence through the Pointwise Mutual Information (PMI) method using a few seed words and detected aspects matching the words surrounding the sentiment words.

In this way, they obtained sentiment words and aspect-sentiment pairs, and then masked and predicted them to calculate separate losses. In particular, for sentiment words, objectives were organized to predict their positivity based on final hidden states by performing word polarity prediction. So the model used three tasks of Sentiment Word Prediction, Word Polarity Prediction, and Aspect-Sentiment Pair Prediction, and the three losses were combined to form the final loss. SKEP reported slight improvements over previous state-of-the-art models and the RoBERTa baseline in sentence- and aspect-level sentiment classification and opinion role labeling tasks.

Like their studies, this dissertation also used the method of infusing sentiment information with BERT. We built sentiment features using a Korean sentiment lexicon and combined the feature embeddings with BERT's input embeddings to train the sentiment-combined models. Although the method of constructing the BERT input embeddings is similar to that of SentiLARE, our models do not include pre-trained sentence-level labels and part-of-speech information. In our models, not only word-level polarity but also intensity are included as sentiment features. We do not predict the infused token-level sentiment values during training processes, so additional prediction tasks or

losses are not used in our models despite the infusion of sentiment features.

The above models include training from scratch, which requires a lot of time and cost. We considered a method that is slightly lighter than the aforementioned models, while also adding features. It combines a sentiment-combined model, trained on a small scale and in small steps, with an existing pre-trained model. Then, in the fine-tuning phase for a task, the two models are optimized simultaneously to perform prediction. It can produce satisfactory task performances while requiring less training time and hardware resources. We will explain this in detail in Chapter 7.

2.4 Chapter Summary

This chapter introduces previous studies on BERT and variations of models based on BERT. First, we provided a rough overview of language models, from traditional n-gram models to contextualized embedding models. We also describe the attention mechanism applied to the language models to improve modeling performances.

The BERT-based models described in 2.2 show a trend towards improving the internal architecture of the model in order to create more effective or efficient models, towards including models which are newly-trained based on non-English languages, and towards some studies attempting to improve or fit the model for specific NLP tasks. Following these lines of development, we also trained and released a new BERT-based model trained using Korean texts.

In 2.3, we roughly explained the studies on sentence-level sentiment classification, since we will provide a model applied to sentiment analysis using sentiment features in this dissertation. We introduced previous works, from the traditional methodology to the recent BERT-based models. Studies which added sentiment features to the BERT architecture provided us with our inspiration to use these additional features in order to improve the performance of related tasks.

3 BERT Architecture and Evaluations

3.1 Bidirectional Encoder Representations from Transformers (BERT)

The Bidirectional Encoder Representations from Transformers model, BERT (Devlin et al. 2019), is the basic model of this dissertation, which trains on the context of sentences bidirectionally to produce generic language representations that can be applied to various NLP tasks. In other words, the model analyzes unlabeled texts for training and obtains the sentence embeddings that reflect meaning and context. Attempts to construct proper text embeddings have long been provided through a variety of methods: by using bag-of-words models, hand-crafted linguistic features, and word embeddings based on recurrent networks.

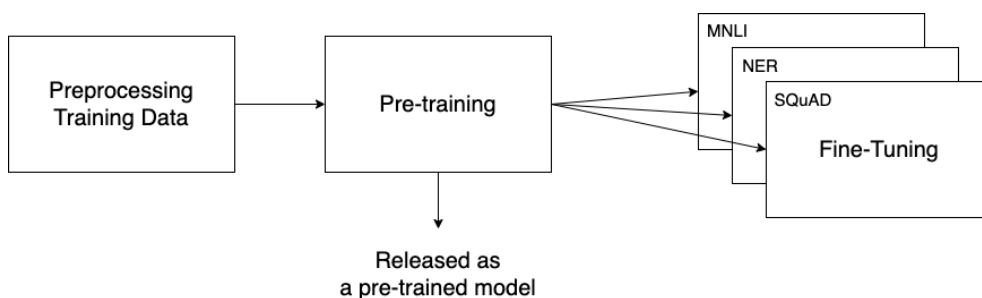


Figure 1: The Process of Training and Fine-tuning BERT

Figure 1 shows the process in which the BERT model is trained and applied to several NLP tasks in the form of a pre-trained embedding model.

BERT builds a model that includes text embeddings through training, and the trained language representations apply just one additional task-specific classifier layer to downstream tasks to adjust the parameters of the model to the task. Through this classifier layer, the sentence embeddings produced by the model are inserted into each task as an input. Accordingly, the model is evaluated on several linguistic tasks on benchmarks such as General Language Understanding Evaluation (GLUE) (Wang, Singh, et al. 2018) and SuperGLUE (Wang, Pruksachatkun, et al. 2019), reporting higher performance than other existing language representation models including ELMo and GPT.

In 3.1, we describe the BERT architecture details: the multi-head self-attention mechanism and the layers used to implement it, the two subtasks used for BERT training, and the fine-tuning process.

3.1.1 Transformers and the Multi-Head Self-Attention Mechanism

BERT was designed based on the Transformer architecture of Vaswani et al. (2017). The Transformer used an attention mechanism to reduce the computations from those of the existing models based on recurrence or convolution, and easily calculate the global dependencies between inputs and outputs.

The Transformer consists of a stack of encoders and decoders, as shown in Figure 2, each of which has multiple stacked blocks, including a multi-head attention layer and a feed-forward network layer. In Figure 2, the block on the left represents the encoders, and the one on the right indicates the decoders. Repetition of the same blocks means that the architecture repeats semantic

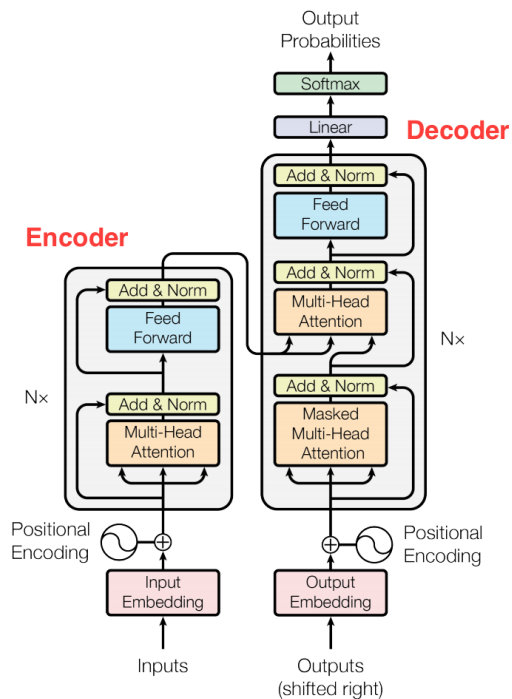


Figure 2: The model architecture of the Transformer (Vaswani et al. 2017)

analysis on the same sequence, so we can expect that it will be better at grasping the complex semantics that the sequence contains. BERT is implemented based on the encoder portion of the Transformer, increasing the number of layers and improving on the internal system, making it more elaborate.

The encoder of Transformer, which is the basis of BERT, has a stack of multiple identical layer blocks, and each layer contains two sub-layers: 1) a multi-head self-attention mechanism layer, and 2) a position-wise fully connected feed-forward network. The former performs the semantic processing of texts, which is the primary function of the Transformer, and the latter trans-

forms the output of the previous encoder layer to the form of the input to the next encoder layer or to the decoder blocks.

First, an attention mechanism, the main part of the Transformer and BERT, calculates the degree to which a token attends to the others in a sequence. Attention is related to the relevance or dependence of tokens, so the more relevant the tokens are, the greater the value of the attention weights.

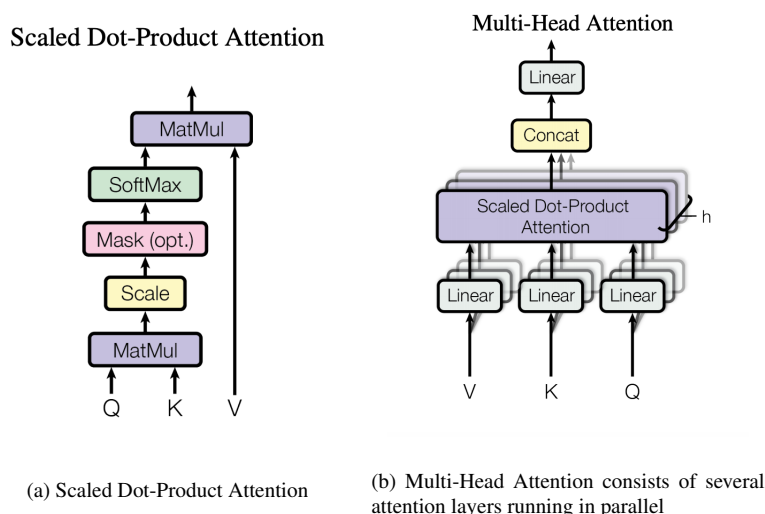


Figure 3: Attention calculation (Vaswani et al. 2017)

To calculate the attention between tokens, three matrices, Query (Q), Key (K), and Value (V), are obtained by a linear transformation of token sequences. A Query is composed of a token in a sequence for finding relevance to other tokens. The remaining tokens in the sequence (apart from the Query tokens) are used to construct Key and Value matrices to calculate the tokens' attention to the Query tokens. The calculated attention functions as an indicator

of finding a token relevant to Query among those tokens. Keys and Values are composed from each token in the sequence, and the former calculates the probability of relevance between the Query token and the remaining tokens, while the latter calculates the attention values based on this probability. These vectors are applied to an attention function using a scaled dot-product, as in Equation (2), which is visualized in Figure 3a:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2)$$

where d_k indicates the dimension of the Key matrix K . This scaled dot-product attention calculates the dot-product of the Query matrix Q and the Key matrix K , scales it by d_k , and then applies softmax to obtain the weights on the Value matrix V . This calculation produces the attention weights between tokens in token sequences. Here, if the token sequences attending to each other are the same, then this process is a self-attention mechanism. In other words, each token in a sequence calculates attention weights with the remaining tokens in the sequence to learn the context information in the sequence.

It is more beneficial to model training using different Q , K , and V matrices obtained by multiple transformations than using only one set of linearly transformed vectors. In other words, the linear projections of Q , K , and V are performed several times differently, and attention functions are applied to each of them in parallel. The calculation according to each linear projection

corresponds with a single attention head, and the multiple outputs obtained by this method are concatenated to become the output of the multi-head attention, as in Figure 3b and Equation (3) (Vaswani et al. 2017).

In Equation (3), W^o indicates the weight matrix that is assigned to each attention head $head_h$ when concatenating the heads to obtain the final representations for a sequence. QW_i^Q , KW_i^K , and VW_i^V represent the weight matrices assigned to the matrices Q, K, and V, where i indicates the calculation in the case of the i^{th} attention head.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^o \\ \text{where } \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \quad (3)$$

The point-wise feed-forward networks, which make up the second sub-layer of the encoder layer of the Transformer, consist of two linear transformations and a ReLU activation between them, as in Equation (4) (Vaswani et al. 2017).

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (4)$$

Here, BERT used GeLU instead of ReLU. This process is applied to each token in each position of a sequence, named point-wise. The output of the previous layer, the multi-head attention layer, is processed into the final output of the block through this feed-forward network layer, which transforms it into

the proper form for the input of the next encoder or decoder block.

The two sub-layers described above are connected after performing residual connections and layer normalization, as in Equation (5) (Vaswani et al. 2017).

$$\text{output} = \text{LayerNorm}(x + \text{SubLayer}(x)) \quad (5)$$

Here, the residual connection process helps the layer preserve previously-learned information and add newly-learned knowledge. Via a simple calculation, it decreases the load of the additional learning of each layer. The layer normalization process ensures that the layer values do not change abruptly, thus speeding up training and improving the generalization of calculated values.

3.1.2 Tokenization and Embeddings of BERT

BERT's input includes sentences in raw texts, which are converted to the token sequences segmented by the WordPiece tokenizer (Schuster and Nakajima 2012; Wu et al. 2016). The WordPiece model was created by the Google Speech Recognition System to resolve the segmentation problems of Korean and Japanese, dealing with rare words and out-of-vocabulary (OOV) words, as the model splits words into WordPiece tokens, subword tokens smaller than words, and uses them as a basic unit. For example, if a word 'embedding' is not included in a model vocabulary, the model can not correctly process the

word. However, if the word is split into WordPiece subword tokens, such as ‘em, ##bed, ##ding’, and those are included in the vocabulary, the tokens can compose their embeddings respectively. Then, the continuous token embeddings construct a context between themselves to process a valid meaning. It would help the model understand and process the meaning of the words and the sequence rather than treating the OOV word with [UNK], an arbitrary value for unknown words.

The WordPiece model first constructs an inventory of separate characters, and then trains a language model with its training data and the inventory to obtain a final vocabulary by combining the characters and tokens in the inventory in a greedy manner, adding them to the inventory. The combined tokens are chosen when the likelihood of the model for the training data increases when the token is added to the inventory. This process is repeated until the pre-defined word limit is reached, or until the likelihood increase falls below a certain threshold.

The raw text input composes a sequence with a single sentence or by concatenating two sentences from a pair according to the data. BERT then adds special tokens to the token sequence for training. [CLS] is added to the front of the token sequence, and the final hidden output of [CLS] includes the compressed information of the sequence for classification tasks. [SEP] tokens are added to the boundaries of sentences within the sequence.

Example (4) is a tokenized sequence processed by a cased BERT-Base model, released by Google Research. Two sentences are concatenated with the

special tokens [CLS] and [SEP]. The tokens with ‘##’ in front of them are the non-first pieces of subword tokens split by the WordPiece algorithm, where the original word was an OOV word. The word ‘shawl’ is not included in the BERT vocabulary, so it is split into three WordPiece tokens, ‘s,’ ‘##haw,’ and ‘##l.’

- (4) original form: Don’t let her get cold, do you hear? Has Heidi a shawl?
 tokenized: [CLS] Don ’ t let her get cold , do you hear ? [SEP] Has Heidi a s ##haw ##l ? [SEP]

The WordPiece tokens in the sequence are assigned integer values according to the index defined in the vocabulary, and converted to integer vectors (token embedding). Also, the learned embeddings indicating segments (boundaries) of sentences are constructed (segment embeddings). If a sequence consists of two sentences, the segment embeddings assign 0 to the tokens in the first sentence, and 1 to the tokens in the second sentence.

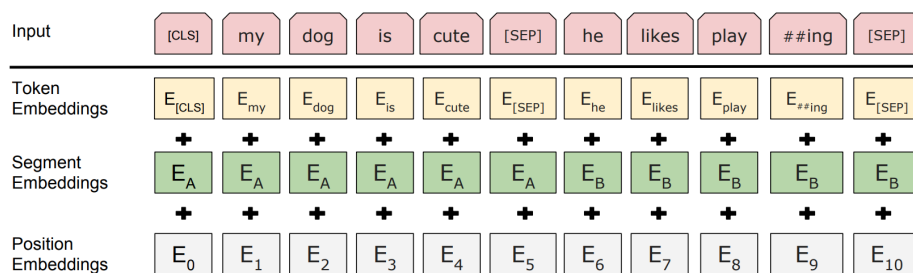


Figure 4: BERT embeddings (Devlin et al. 2019)

Additionally, since the self-attention mechanism does not include the position information of tokens in a sequence, positional embeddings are constructed to represent each token’s relative position (position embedding). The positional encoding of each token position is performed through sine and cosine functions. BERT’s input embeddings are constructed by point-wise summing the three embeddings described above, as visualized in Figure 4.

3.1.3 Training and Fine-Tuning BERT

The original English-based BERT model was trained based on a large-scale corpus, including the BookCorpus (Zhu et al. 2015) and English Wikipedia texts. The texts are converted to embeddings as described in 3.1.2 and trained through two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

MLM is a task for a bidirectional language modeling of tokens in a sequence. BERT masks a randomly chosen 15% of tokens from each sequence and predicts them by language modeling through this task. As an autoencoder language model, with final hidden states corresponding to the masked tokens, BERT calculates its cross-entropy loss through softmax for all the candidate tokens in its vocabulary to predict the original tokens, as in Equation (6).

$$-\frac{1}{MN} \sum_{n=1}^N \sum_{k=1}^K \sum_{m=1}^M \left(y_{m,n}^k \times \log(\text{softmax}(x_{m,n}, k)) \right) \quad (6)$$

In Equation (6), M and N represent the number of masked tokens and the

number of sequences in the data, respectively. K indicates the number of labels, that is, the candidate tokens in the vocabulary for the masked tokens. $x_{m,n}$ represents the input embeddings for the m^{th} masked token in the n^{th} sequence in the training data. $y_{m,n}$ indicates the target token label for the m^{th} masked token in the n^{th} sequence in the training data.

However, masking tokens in the training data could create a gap from task data whose tokens are not masked. Therefore, BERT masks only 80% of the tokens chosen to be masked, replaces 10% of them with random tokens in the vocabulary, and uses the unchanged tokens for the remaining 10%. Through this process, the model learns context information composed of linking tokens within a sequence.

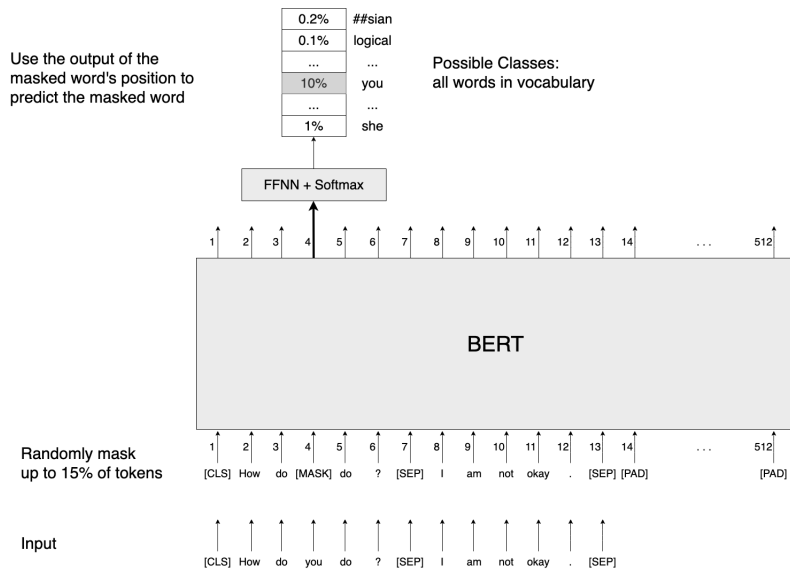


Figure 5: Visualization of Masked Language Modeling

Figure 5¹ represents the MLM task. In the figure, the input sequence says, ‘[CLS] How do you do ? [SEP] I am not okay . [SEP],’ and the token ‘you’ is randomly chosen to be masked. This model has set its max sequence length to 512, and the remaining positions are padded with the [PAD] tokens. BERT estimates probabilities for all the tokens in its vocabulary to be the correct answer of the masked token. The upper part of the figure shows selecting the token ‘you’ as the prediction from the vocabulary candidates since it has the highest probability.

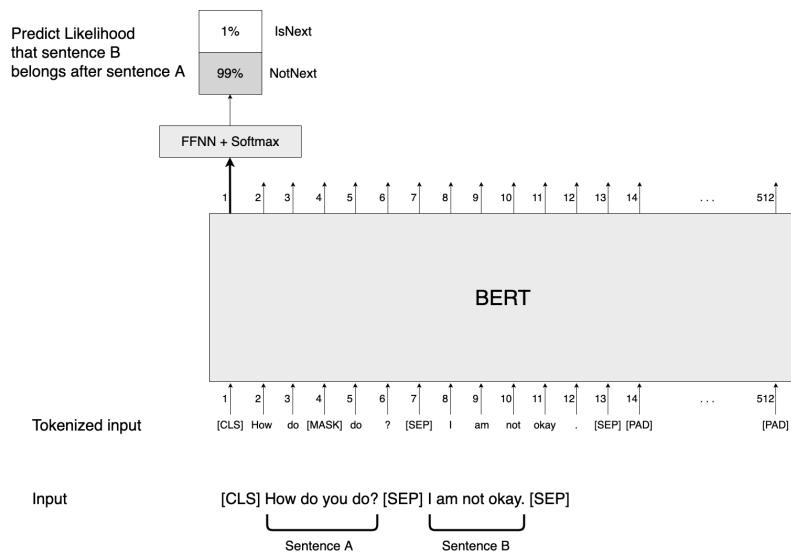


Figure 6: Visualization of Next Sentence Prediction

On the other hand, NSP is a task designed for learning the relations between two sentences in a sequence, which cannot be captured by language

¹Referring to <http://jalammar.github.io/illustrated-bert/>.

modeling in sequences directly. BERT composes a sentence pair by extracting two sentences from each example from its corpus. Here, assume the first sentence in the sequence as ‘sentence A’ and the second one as ‘sentence B’ for convenience. For each sentence A, 50% of the B sentences are chosen as the sentence immediately after A in the document, and 50% are chosen to be a random sentence from the corpus. The model then predicts whether sentence B is the next sentence of A or not. The hidden states of [CLS], which is the first token of the sequence, are used for NSP prediction, as described in Figure 6². The upper part of the figure shows the prediction of ‘NotNext’, with a probability of 99% that the two sentences are not sequential.

For the NSP task, BERT calculates binary cross entropy loss with the hidden states of input sequences, as in Equation (7).

$$-\frac{1}{N} \sum_{n=1}^N [y_n \times \log(\text{softmax}(x_n)) + (1 - y_n) \times \log(1 - \text{softmax}(x_n))] \quad (7)$$

In Equation (7), N represents the number of sequences in the data. x_n represents the input embeddings for the n^{th} sequence in the training data, and y_n indicates the target label of IsNext or NotNext for the n^{th} sequence in the training data.

The model calculates losses of the two tasks above and is trained to reduce them. Here, loss is an indicator of the difference between the correct answer and the prediction. The smaller the number, the closer the answer and predic-

²Referring to <http://jalammar.github.io/illustrated-bert/>.

tion are, which means a better prediction. The model reduces this loss value to a certain convergence level by repeating the training process for several steps. The training performance of the BERT model is expressed in the accuracies and losses of MLM and NSP. In particular, the global loss of the BERT training process is obtained by the sum of the MLM loss and NSP loss.

Through these tasks, the model learns various linguistic knowledge, including the lexical, syntactic, semantic, and contextual information of texts, through the internal stacked layers as described in 3.1.1. This contextualized embedding model captures information in narrow span contexts such as words at lower layers, and information of wider span such as the relationship between words at higher layers. The model is expected to learn linguistic knowledge while not using hand-crafted features such as sentence length, word frequencies, and capital letters, which were utilized by traditional approaches.

Various sources have released BERT-like pre-trained models. Google Research provides their TensorFlow-based models at their GitHub³, and Wolf et al. (2020) collected several released models in the form of a package in a library called Transformers, based on PyTorch. Appropriate task-specific inputs and outputs are applied to those models for any downstream task to fine-tune the models and produce the task results.

The pre-trained models are loaded and further trained for a few steps based on task datasets to adjust model parameters to fit the task by fine-tuning. Here, the task datasets include labels to be predicted by the fine-tuned models.

³<https://github.com/google-research/bert>

A task-specific classifier layer is implemented and applied to the pre-trained models. For example, BERT’s output representations are extracted in sequence form and applied to token-level tasks such as sequence tagging or Question Answering at the task-specific classifier layer. In the classification of sentiment or relations of sentences, only the embeddings of the [CLS] token from a sequence are extracted. Then the model calculates task-specific prediction losses using the output representations and the gold-standard labels from the datasets.

This fine-tuning process is very inexpensive compared to training. Training can require several days depending on the number of training steps, even when using a Cloud TPU, while fine-tuning takes only a few hours with GPUs.

3.2 Evaluation of BERT

3.2.1 NLP Tasks

Language models, including BERT-based models, are evaluated by several NLP tasks related to various linguistic phenomena. Since studies on language modeling have focused on English corpora, most of the NLP tasks released consist of English texts. For non-English languages, most task datasets are composed, collected and annotated similar to those of English tasks. Therefore, we here introduce major English NLP tasks and their datasets, including those used for BERT evaluation.

- Acceptability Task: Models predict a binary grammaticality label (gram-

matically or ungrammatically) for each English sentence extracted from published linguistics literature. The Corpus of Linguistic Acceptability (CoLA) constructed by Warstadt et al. (2019) is commonly used.

- Question Answering
 - The Quora Question Pairs (QQP) (Iyer et al. 2017): Models process pairs of potential question duplicates and predict if the two questions are semantically equivalent or not.
 - The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al. 2016): This corpus consists of question-answer pairs, in which the question is created based on Wikipedia articles and the answer is found from the article. Models predict whether the pair of question and answer is correctly matched.
 - Question Natural Language Inference (QNLI) corpus: This is a modified version of SQuAD, including annotated question-paragraph pairs, for which models predict whether the paragraph contains the answer for the question.
- Sentiment Analysis: Models predict sentiment labels of given sentences or documents. The Stanford Sentiment Treebank (SST) (Socher et al. 2013) includes movie review texts which are classified as positive or negative (SST-2) or categorized by five refined classes of subjectivity (SST-5).
- Paraphrase Detection: With the Microsoft Research Paraphrase Corpus

(MRPC) (Dolan and Brockett 2005), models judge whether sentence pairs of the dataset extracted from news sources are semantically equivalent or not.

- **Natural Language Inference (NLI):** Sentence pairs composed of a premise and a hypothesis sentence are annotated with entailment relations between them, and models predict whether the premise entails or contradicts with the hypothesis or neither. For this task, several datasets are released: The Stanford Natural Language Inference (SNLI) (Bowman et al. 2015), The Multi-Genre Natural Language Inference Corpus (MNLI) (Williams et al. 2018), and The Cross-Lingual NLI Corpus (XNLI) (Conneau et al. 2018).
- **Recognizing Textual Entailment (RTE):** The task dataset is constructed based on news articles and Wikipedia texts, and models predict whether two sentences in a pair are in an entailment relation, a paraphrase relation, or neither. The GLUE benchmark evaluates model performances with the RTE datasets used over several years together: RTE1 (Dagan et al. 2005), RTE2 (Bar-Haim et al. 2006), RTE3 (Giampiccolo et al. 2007), and RTE5 (Bentivogli et al. 2009).
- **Semantic Textual Similarity (STS):** This task includes scoring the similarity between two sentences extracted from image captions, news headlines, and user forums. The similarity score ranges from 0 (completely dissimilar) to 5 (completely equivalent), indicating a semantic distance

between two sentences' language representations. Datasets from SemEval 2012-2017 are commonly used.

- **Named Entity Recognition (NER):** Models predict entity types for each token in a sequence from a news article corpus, for example, the name of a person (PER), place (LOC), or organization (ORG). The CoNLL 2003 NER task dataset (Sang et al. 2003) is usually used.

Many of these tasks are also provided by benchmarks such as GLUE (Wang, Singh, et al. 2018) for easy model evaluation. Referring to these tasks or directly from these sources, non-English task datasets have been constructed. Such tasks are also being provided in Korean: Question Answering (KorQuAD), Natural Language Inference (KorNLI), Semantic Textual Similarity (KorSTS), Named Entity Recognition (Naver-NER), Movie Review Classification (NSMC), and Hate Speech Detection (HSD). Some datasets have been collected and evaluated by each researcher, such as movie review datasets from other sources and cosmetics product reviews, and were introduced by papers but have not yet been released. These Korean tasks are described in detail in Chapter 5, evaluating the performance of Korean pre-trained models.

3.2.2 Metrics

The evaluation metrics commonly used in these NLP tasks include accuracy and F1 scores, which predict classification performance. Specific metrics may be used according to the task, but accuracy and F1 score are usually used in most tasks. It is necessary to consider whether the test dataset's actual answers

match the model’s classification results to explain these two metrics. Table 1 represents a confusion matrix that is composed of the classification of the model and the actual answers of the dataset.

Table 1: Confusion Matrix

		Actual Answers	
		TRUE	FALSE
Model Classifications	TRUE	True Positive (TP)	False Positive (FP)
	FALSE	False Negative (FN)	True Negative (TN)

From the confusion matrix, four combinations of classification results and actual answers can be identified as below, of which TP and TN are correct cases, and FP and FN are incorrect cases of the model prediction.

- True Positive (TP): The model predicts TRUE, and the actual answer is TRUE. (correct)
- False Positive (FP): The model predicts TRUE, and the actual answer is FALSE. (incorrect)
- False Negative (FN): The model predicts FALSE, and the actual answer is TRUE. (incorrect)
- True Negative (TN): The model predicts FALSE, and the actual answer is FALSE. (correct)

Based on the cases above, precision and recall are calculated to produce an F1 score and accuracy, as in Equations (8-11). The F1 score is calculated by the harmonic mean of precision and recall (Equation (10)), in which precision indicates the proportion of True Positive cases out of the total positive cases which are predicted by the model as TRUE (Equation (8)). Recall represents the proportion of True Positive cases out of the actual TRUE cases annotated as gold labels, as in Equation (9).

$$Precision = \frac{TP}{TP + FP} \quad (8)$$

$$Recall = \frac{TP}{TP + FN} \quad (9)$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (10)$$

Accuracy indicates the proportion of correct predictions of the model out of all cases, as in Equation (11).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (11)$$

3.3 Chapter Summary

This chapter describes the BERT model and the encoder part of the Transformer, which is the basis of the model proposed in this dissertation, explain-

ing the multi-head self-attention mechanism that forms its foundation. Moreover, we mentioned BERT's input text form and tokenization process and explained the construction of embeddings based on the tokenized units. We also summarized BERT training and fine-tuning according to each NLP task using such embeddings.

In 3.2, we referred to various NLP tasks and their performance as indirect indicators of the BERT model's language processing capabilities. We mentioned major English NLP tasks, which are the starting point for many non-English studies. Additionally, the universal measures of assessing such tasks, accuracy, and F1 scores were described based on the confusion matrix, which illustrates model predictions with the correct answers from task datasets.

4 Pre-Training of Korean BERT-based Model

This dissertation proposes a monolingual BERT-based model trained based on Korean materials. The first released BERT model that handled Korean texts was Google Research’s multilingual BERT (M-BERT). It is a model that has trained with texts written in 104 languages, including Korean, English, German, Arabic, and Slovenian, and a vocabulary constructed from them, being able to process texts in all the languages included in the model.

However, this multilingual model has shortcomings when compared to monolingual models, despite the advantages of its multilingualism. The disadvantages of the multilingual model are described in 4.1. This study attempted to construct a Korean-specific vocabulary and language model so that linguistic knowledge in Korean texts can be better captured to mitigate those shortcomings. Thus, in experiments observing the models’ performance in this dissertation in Chapter 5, M-BERT was used as a baseline model.

The main content of this chapter is based on our existing work (Lee, Jang, et al. 2020a,b).

4.1 The Need for a Korean Monolingual Model

First of all, M-BERT tried to deal with texts in 104 languages at a time and composed its training data only with Wikipedia texts written in those languages. The disadvantage of this is that the domain of the training data is limited to an online encyclopedia. Thus, its linguistic characteristics, including words and writing styles, differ from texts of other domains such as books,

blog posts, and comments. Training a model with limited data may not sufficiently capture and learn common, various linguistic phenomena. It would be beneficial to set up various domains and sources for training data to construct a generic pre-trained language model. Usually, monolingual models, including English, organize training data with texts from various domains such as news articles and books. We also attempted to compose training data from diverse domains to train Korean-specific models.

The vocabulary of M-BERT contains tokens from 104 languages, while not including enough numbers and types of tokens of each language. For instance, M-BERT's cased model ("BERT-Base, Multilingual Cased") has 119,547 tokens in its vocabulary, including in that only 3,273 Korean tokens (2.74%). The vocabulary covers only 1,187 Korean syllable characters. Therefore it is hard to judge that Korean texts can be tokenized and processed by M-BERT properly. In this case, many words will be processed as out-of-vocabulary (OOV) words and will be assigned random values in the corresponding embeddings.

For example, M-BERT processes Korean words **멧사람** paytsalam "seaman" and **춡다** chwupta "cold" as OOV words since its vocabulary does not include the words. The model could not perform subword tokenization for the words since it does not have the syllables **멧** payt and **춡** chwup in its vocabulary. However, such phenomena are not likely to occur in monolingual models.

In particular, since Korean is an agglutinative language, various word

forms are created depending on the combination of word roots, case markers, and affixes, much more than other languages such as English, which is a fusional language. Moreover, the word forms are highly likely to change their form since a combination of graphemes constructs Korean syllables. For example, the forms 온다 onta “come,” 왔다 wassta “came,” 와서 wase “come,” and 올 이 “to come” have different shapes in the form of characters. ‘온 on, 왔 wass, 와 wa, 올 이’ all include the meaning of ‘come,’ but in BERT’s vocabulary, it has no choice but to treat them as separate entries.

However, it is not easy to include all forms in a vocabulary. The number of tokens in the vocabulary will grow indefinitely, considering both the large number of words and their various forms. If the vocabulary size is too large, the number of candidate tokens to be predicted in Masked Language Modeling during model training grows, requiring longer training time and hardware cost, depending on the computation workload.

Furthermore, the meaning of words and sentences will not be appropriately processed unless words are split into correct subword units because of inadequate vocabulary composition. For example, when the word 마셨다 masyessta “drank” is tokenized, if the tokens 마셨다 masyessta or 마셨 masyess “drank” and ##다 ##ta (suffix) are not included in the BERT vocabulary, this word is split into subwords ‘마 ma, ##셨 ##syess, ##다 ##ta.’ If even the three subwords are not included in the vocabulary, the word is processed as an OOV word. The subword sequence ‘마 ma, ##셨 ##syess, ##다 ##ta’ or the OOV-processed value [UNK] is unlikely to be understood as ‘drank’ because

the form of 마셨- masyess-, the minimum unit which contains the meaning of ‘drank’, is not maintained. Although the sequential subword tokens are contextually processed to achieve some meaning, which is an advantage of the WordPiece tokenizer of BERT, it is hard to find the unique meaning of each split token. Of course, in the case of the [UNK]-processed token, it is harder to guarantee that it maintains its original meaning.

Therefore, when implementing a Korean-specific model, it is necessary to properly include small subword units to resolve the OOV problem, and larger word units to maintain word meaning in its vocabulary at a realistic rate. Several pre-trained Korean-specific models are now available, and like most of them, we assumed appropriate vocabulary size limits and constructed a vocabulary based on the WordPiece tokenizer. Thus, the vocabulary consists of both single characters and concatenated characters as a subword and word unit.

4.2 Pre-training Korean-specific BERT Model

In this dissertation, we trained two Korean-specific BERT models, KR-BERT and KR-BERT-MEDIUM, based on different compositions of training data and vocabulary. We performed training of these models with the help of a study (NRF-2017M3C4A7068186) supported by the National Research Foundation of Korea (NRF), in which graduate students in our laboratory performed the preprocessing of training data and vocabulary construction. Both of the two models use character-level vocabularies based on the WordPiece to-

kenizer. Processing texts based on morpheme-level vocabularies and models such as HanBERT and the morpheme-based version of KorBERT may help reflect the agglutinative property of Korean better. However, in that case, all the texts used for model training and task-specific evaluation should be tokenized using external part-of-speech taggers, which is very costly in processing time and hardware requirements. Therefore, we decided to utilize character-level tokens for our models in this dissertation.

For model training, two tasks of Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), are used as provided by BERT. By MLM, the model learns contextual information composed of relations between tokens in a sequence, and by NSP, it learns the context based on the relations between sentences in a document.

The training data and vocabulary of each model we implemented are as follows.

KR-BERT

KR-BERT (**Ko**R**e**an-based BERT pre-trained model) is the main monolingual model we provide. For this model, we extracted texts from the Korean Wikipedia dump¹ using WikiExtractor², omitting metadata and history for this model. We also extracted crawled news articles, such as Chosun Ilbo, JoongAng Ilbo, and Kookmin Ilbo. The total data size is 2.47GB, which consists of 20M sentences and 233M words.

¹<https://dumps.wikimedia.org/kowiki/latest/>

²<https://github.com/attardi/wikiextractor>

The vocabulary of this model includes 16,424 tokens that are organized based on this training dataset. We first collected tokens for our vocabulary by training with the WordPiece model and refining it by uniting different trained vocabularies to obtain a better and more diverse list of tokens. Then we chose the list of 10,256 tokens by comparing several models trained using different vocabularies for 100,000 steps. A model with the vocabulary of 10,256 tokens produced the best MLM performance among various vocabulary sizes, as shown in Table 2.

Table 2: Comparison of MLM performance for KR-BERT according to vocabulary size

Vocabulary size	6,713	8,353	10,256
MLM loss	2.9182556	3.0208275	1.8139153
MLM accuracy	46.80	46.94	63.04

To the 10,256 tokens in the vocabulary, we heuristically added special symbols frequently used in user-generated Korean textual data and several other languages (Latin alphabet, Japanese Kana, and Chinese characters) to include 16,424 tokens. The main work of data preprocessing and vocabulary construction was carried out by graduate students in our laboratory as part of the study supported by the NRF.

This model is released on GitHub³ and can be publicly accessed.

³<https://github.com/snunlp/KR-BERT>

KR-BERT-MEDIUM

We experimentally trained an additional model, KR-BERT-MEDIUM, with the training data expanded from those of KR-BERT. We trially trained an additional model, KR-BERT-MEDIUM, with the training data expanded from those of KR-BERT. Here, the model name has a suffix ‘MEDIUM’, since its training data are larger than KR-BERT’s original dataset. We have another additional model, KR-BERT-EXPANDED, for our NRF study, with more extensive training data expanded from that of the KR-BERT-MEDIUM model, so the suffix ‘MEDIUM’ is used.

The expanded training data includes Korean Wikipedia, news articles, legal texts crawled from the National Law Information Center, and the Korean Comments dataset⁴. This data expansion composes the more extensive training set with more various domains than those of KR-BERT. The total data size is about 12.37GB, consisting of 91M and 1.17B words.

This model’s vocabulary size is 20,000, whose tokens are trained based on the expanded training data using the WordPiece tokenizer. The size is also chosen by comparing different vocabulary sizes by training separate models with them for 100,000 steps, and picking the one with the higher MLM performance, as shown in Table 3.

The main work of data preprocessing and vocabulary construction was carried out by graduate students in our laboratory as part of the NRF study.

⁴https://github.com/Beomi/KcBERT/releases/tag/TrainData_v1

Table 3: Comparison of MLM performance for KR-BERT-MEDIUM according to vocabulary size

Vocabulary size	20,000	30,000
MLM loss	3.53247164	3.7084162
MLM accuracy	39.48	38.13

Table 4 represents the composition of the vocabularies of KR-BERT and KR-BERT-MEDIUM compared to that of other existing models. Here, only BERT-based and character-level models that can deal with Korean texts were chosen to be compared.

Table 4: Composition of the vocabularies

	M-BERT	KorBERT	KoBERT	KR-BERT	KR-BERT-MEDIUM
words (Hangul)	1,664 (1.39%)	12,047 (39.12%)	4,489 (56.10%)	7,352 (44.76%)	10,585 (52.93%)
subwords (Hangul)	1,609 (1.35%)	8,023 (26.05%)	2,678 (33.47%)	3,840 (23.38%)	6,203 (31.02%)
symbols and other languages	116,170 (97.18%)	10,722 (34.82%)	830 (10.37%)	5,227 (31.83%)	3,207 (16.04%)
special tokens	5 (0.004%)	5 (0.02%)	5 (0.06%)	5 (0.03%)	5 (0.03%)
total	119,547	30,797	8,002	16,424	20,000

In the table, words and subwords are written in Hangul, in which a whole word form is mapped to one token (for example, 가방 kapang “bag” and 멋진

mescin “wonderful”) and a subword is composed of more than one syllables split in a word (for example, ##가 ##ka (subject case marker) and ##개월 ##kaywel “month(s)”). The group of symbols and other languages includes Latin alphabets, Japanese Kana, Chinese characters, special symbols, and emoticons used in Korean texts. Finally, the special tokens indicate the five tokens used by BERT while text processing: [CLS], [SEP], [MASK], [UNK], and [PAD].

M-BERT included only a small number of tokens for each language to cover tokens from 104 languages in a limited-sized vocabulary. So the proportion of Korean characters in that vocabulary is tiny. Other Korean monolingual models allocate most of the entries in their vocabularies to Hangul tokens, mitigating the disadvantage of M-BERT’s vocabulary composition.

KR-BERT has a vocabulary composition similar to other Korean models, especially KorBERT, and includes many more symbols and tokens from foreign languages than KoBERT. This vocabulary will help the trained models perform NLP tasks containing texts from various domains with different stylistic properties. User-generated web data such as comments, writings on online forums, or blog posts are likely to include foreign languages, special symbols, and emoticons.

The KR-BERT-MEDIUM model’s vocabulary contains a larger proportion of Hangul word tokens than KorBERT and KR-BERT, and more Hangul word tokens than KoBERT despite its lower proportion of Hangul words. In general, it can be expected that the proportion of words that are not split into

subwords and processed as a token itself is high. There will be many varied word forms, such as noun + case marker and root + suffix as those tokens. As examples in 4.1 show, words such as 마셨다 *masyessta* “drank” and 마시는 *masinun* “drinking” are not split into ‘마 *ma* ##셨 *##syess* ##다 *##ta*’ and ‘마 *ma* ##시 *##si* ##는 *##nun*’ and are instead processed as single tokens themselves. Each token will better keep its original meaning, helping the model analyze the meaning of a sequence from the word meaning to the context of the overall sequence. On the other hand, the ratio of symbols and foreign characters is low for the size of the vocabulary. Even still, it is not a small quantity.

Additionally, in Table 5, we present the vocabulary sizes, parameter sizes, and data sizes for our models, comparing with M-BERT and several previous studies on Korean pre-trained BERT models.

Table 5: Details of Korean pre-trained BERT models

	M-BERT	KorBERT	KoBERT	KR-BERT	KR-BERT-MEDIUM
vocabulary size	119,547	30,797	8,002	16,424	20,000
parameter size	167,356,416	109,973,391	92,186,880	99,265,066	102,015,010
data size	- (The Wikipedia data for 104 languages)	23GB	- 25M sentences, 324M words	2.47GB 20M sentences, 233M words	12.37GB 91M sentences, 1.17B words

KR-BERT attempted to reduce computations by using smaller training data and vocabulary size, resulting in the parameter size of 99M, the smallest after that of KoBERT. This model’s training data is very small-scaled,

at 2.47GB, about 1/10 of those of the other models. Therefore, it takes less time to convert raw text files into tokenized forms for BERT training, since processing time is proportional to the training data's size. However, its performance is not lower than other models, and even higher in a few tasks. This will be mentioned again in Chapter 5 through task performances.

In comparison, as a trial to improve performance, KR-BERT-MEDIUM has collected more text from more diverse domains as training data. Thus, the training data grew to 12.37GB and the parameter size increased to 102M, similar to that of the other models. Although tokenizing time and the number of computations have increased as the data has grown, the expanded model recorded better task performances, as reported in Chapter 5. Various domains and a sufficient amount of training data may have had a positive effect on model training.

The two models we implemented in this dissertation are trained on the same "Base" scale as the original English-based BERT models. The Base-scale models include 12 encoder layers, 12 attention heads, and a hidden size of 768. That is, 12 different attention heads calculate the attention relations between tokens in a sequence separately, and the concatenated results from all the heads compose the 768-dimensional embeddings. This calculation is repeated through 12 layers of computing and updating the attention values. Through this process, the models learn various linguistic knowledge contained in the texts, including lexical, syntactic, semantic, and contextual information.

KR-BERT is trained for 2M steps with a maximum sequence length (maxlen) of 512, training batch size of 64, a learning rate of 1e-4, and, following the original BERT models, uses the Adam optimizer. It took 79.5 hours to train the model using a Google Cloud TPU v3-8.

Maxlen is the upper limit of sequence length for BERT training. The tokenized sequence is truncated if it is longer than the upper limit, and it is padded by [PAD], a special token of BERT, when it is shorter than maxlen. This is for convenience for when the sentence vectors of multiple layers inside the model are calculated in matrix form. Padded tokens are masked when calculating attention weights; thus, they do not affect performance much. However, if the maxlen is too large, it will affect training speed.

The maximum maxlen value provided by BERT is 512, which we used for KR-BERT, with 99.99% of sentences in the training data being shorter than this maxlen. We set the max predictions per sequence to 77, which is 15% of 512, following the original BERT model. This indicates how many tokens per sequence are masked to be applied to the MLM task.

The batch size represents how many sentences are processed at a time, which can adversely affect performance if too small. However, setting it too large can make hardware memory unaffordable and adversely affect model performance. Therefore, it is set to 64, which is appropriately sized and manageable in our environment.

The steps indicate the units by which model parameters are updated through information learned in a batch. If this is set larger, the parameter values of the

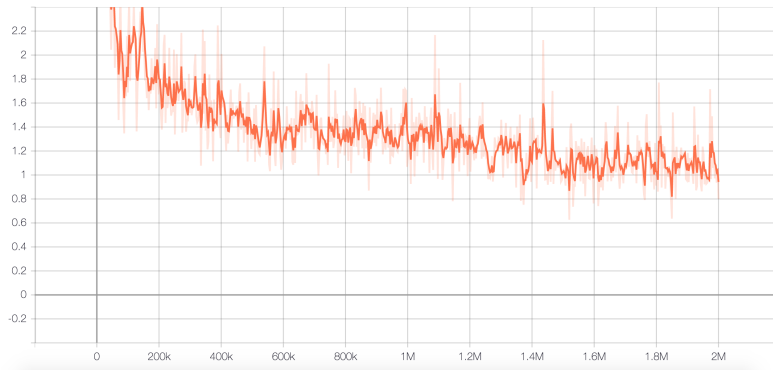


Figure 7: Training loss of KR-BERT

model will be updated more times. In this study, the number of steps was set to the point when training performance (losses and accuracies of training tasks) became stable without much change. Figure 7 visualized the change of loss, a linear combination of MLM and NSP losses, as the model’s global training performance, according to steps. In the figure, the x-axis represents the number of steps, and the y-axis indicates training loss.

KR-BERT-MEDIUM is trained for 2M steps with a maxlen of 128, training batch size of 64, learning rate of $1e-4$, and using the Adam optimizer, taking 22 hours to train the model using a Google Cloud TPU v3-8. Here, as the maxlen was set to 128, the training time was significantly reduced. 99.01% of the training data sentences were shorter than this maxlen, so we judged that truncating the remaining 0.99% of the sentences would cause no issues.

Figure 8 represents the training loss of KR-BERT-MEDIUM. The x-axis represents the number of steps, and the y-axis indicates the training loss. The

sudden drop around the point of 1.5M steps is not exactly explainable, but it is likely to have accidentally occurred in the process of stopping and resuming training for an intermediate check.

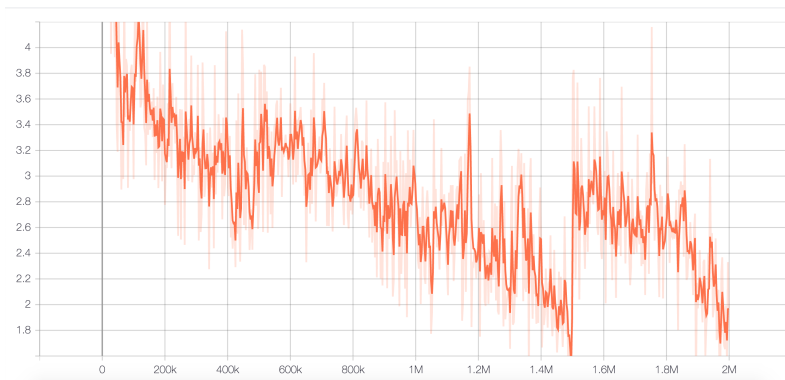


Figure 8: Training loss of KR-BERT-MEDIUM

Table 6: Training performances of KR-BERT and KR-BERT-MEDIUM

	KR-BERT	KR-BERT-MEDIUM
global steps	2M	2M
loss	0.9154538	2.6140482
MLM accuracy	77.34	51.75
MLM loss	0.9511529	2.4622557
NSP accuracy	99.50	89.13
NSP loss	0.013327285	0.2479853

Table 6 shows the detailed training performances of KR-BERT and KR-BERT-MEDIUM. The loss in the table indicates global loss, which is a combination of MLM and NSP losses. KR-BERT is supposed to have adapted well to the training data since its size is small (2.47GB), resulting in high MLM

accuracy 77.34 and NSP accuracy 99.50 during the training process.

The training data of KR-BERT-MEDIUM are relatively large-scaled and composed of texts of various domains and styles, and this could have resulted in lower training performances, especially in MLM. The MLM accuracy of the model is 51.75, and the NSP accuracy is 89.13, which are much lower than those of KR-BERT. However, as shown in Figure 8, KR-BERT-MEDIUM's training performance had already converged without clear improvement before reaching 2M steps. Thus, the training process was completed at 2M steps since the performance gain is small for the effort and resources required for further training.

However, even though training performance was not as high for the data-expanded model, this did not prove to be fatal when applied to actual NLP tasks. In a few tasks, it performed even better than other models. Although it does not achieve very high performance for its training data, its task performances in Chapter 5 were not inferior. Conversely, if the model training performance is too high, task performance may actually prove to be low. The problem of overfitting can occur when models become too familiar only with their training datasets. Therefore, an assessment of the model will have to be done through considering performance in diverse areas, not only through training subtasks but also through several NLP tasks that reflect various linguistic phenomena. We evaluate how well the models trained in this dissertation deal with Korean NLP tasks in Chapter 5.

4.3 Chapter Summary

This chapter first mentioned the shortcomings of the M-BERT model as a multilingual model and explained the need for a monolingual model, especially a Korean-specific language model. The shortcomings of M-BERT include that its training data is limited to one domain, and its vocabulary is not appropriate to capture the linguistic characteristics of each language it covers. We implemented two Korean monolingual BERT models to mitigate these shortcomings.

The second part described the Korean language models KR-BERT and KR-BERT-MEDIUM implemented by this dissertation, and compared the details of these models and their training process to the existing pre-trained models. KR-BERT is trained based on smaller training data and vocabulary than other models, and KR-BERT-MEDIUM is a derivative model of it using an expanded large-scale training dataset and different vocabulary. We also reported the training performances of the models.

5 Performances of Korean-Specific BERT Models

5.1 Task Datasets

Many of the NLP tasks used to evaluate Korean language models include Korean texts and task-specific labels reflecting various linguistic phenomena, similar to the English tasks described in 3.2. Datasets such as Naver-NER for Named Entity Recognition, KorQuAD for Question Answering, KorNLI and KorSTS for Natural Language Inference and Semantic Textual Similarity, respectively, and corpora for Paraphrase Detection and Sentiment Analysis have been provided. Although not many datasets are publicly provided in some tasks, many other datasets are collected and annotated on their own in several studies.

We introduce the tasks we used in this dissertation to evaluate the models we trained in Chapter 4.

5.1.1 Named Entity Recognition

In Named Entity Recognition (NER), named entities in a sequence are detected, and their categories are classified. ‘Named entity’ usually refers to an expression with a unique meaning, consisting of one or more words combined, such as personal names, organization names, and numbers. The Naver-NER dataset¹ is mainly used for Korean NER, which was released at the Naver NLP Challenge 2018, hosted by Changwon National University and Naver. The category of named entities, which is the classification target, is

¹http://air.changwon.ac.kr/?page_id=10

divided into 14 categories, as in Table 7.

Table 7: The category of the named entities

Named Entity Category	Tag	Definition
PERSON	PER	A person’s name (real, imaginary, etc.)
FIELD	FLD	A field of study, theory, law, etc.
ARTIFACTS_WORKS	AFW	Artificial objects created by human
ORGANIZATION	ORG	Institutions, organizations, meetings, and talks
LOCATION	LOC	Regional name, administrative district name, etc.
CIVILIZATION	CVL	Terms related to civilization and culture
DATE	DAT	Date
TIME	TIM	Time
NUMBER	NUM	Numbers
EVENT	EVT	A specific event, accident, event, etc.
ANIMAL	ANM	Animals
PLANT	PLT	Plants
MATERIAL	MAT	Metals, rocks, chemicals, etc.
TERM	TRM	General terms including medical terms, IT terms, etc.

Since the test set is not released publicly, we used the newly-split dataset of the original training set of 90,000 sentences: 81,000 sentences for training and 9,000 sentences for testing models. We referred to a released data split and fine-tuning codes from a GitHub source of Park (2020b), fine-tuning our models and baseline models, and comparing their F1 scores with those reported by KoBERT. Since the fine-tuning code is based on PyTorch, the TensorFlow-based model checkpoints trained in this dissertation were converted to PyTorch models.

5.1.2 Question Answering

For Question Answering, a dataset named KorQuAD 1.0², created by LG CNS AI / Big Data Research Center for Korean Machine Learning Comprehension, is commonly used. It is constructed in the same way as that of Stanford Question Answering Dataset (SQuAD) v1.0, consisting of 10,645 paragraphs and 66,181 question-answer pairs from 1,560 Wikipedia articles. In each pair, one sentence from a paragraph mapped with a question becomes the answer to the question. This task is assessed by Exact Match (EM) and F1 score. EM indicates the rate at which a model correctly matches the actual answers, not counting punctuation and articles. It is marked as ‘1’ if the predicted label and the correct answer are precisely equal and ‘0’ otherwise. The F1 score functions as a partial score, comparing the model predictions and actual answers in syllables, taking into account the overlap of the answers.

The task dataset includes 60,407 question-answer pairs for the training set and 5,774 pairs for the validation set. The test set is not publicly released, so evaluation is performed based on the validation set. The performances of other existing pre-trained models such as KoBERT and M-BERT performed by Park (2020a) are also reported based on the validation set. Referring to the fine-tuning code from the source, we executed the assessment process producing EM and F1 score results. Since the fine-tuning code is implemented based on PyTorch, the TensorFlow-based model checkpoints constructed in this dissertation were converted to PyTorch models.

²<https://korquad.github.io/category/1.0-KOR.html>

5.1.3 Natural Language Inference

As part of the Natural Language Understanding (NLU) study, task datasets for Natural Language Inference (NLI) include sentence pairs composed of a premise and a hypothesis sentence with annotated textual entailments labels. Based on that, models predict whether the premise entails the hypothesis, contradicts it, or neither. Thus, the model predicts one of the three labels: ‘entailment’, ‘contradiction’, or ‘neutral.’

- Entailment: Given a premise, the hypothesis is true.
- Contradiction: Given a premise, the hypothesis is false.
- Neutral: Undetermined

For Korean NLI, the KakaoBrain (Ham et al. 2020) benchmark dataset for Korean Natural Language Understanding provides an NLI dataset named KorNLI. This dataset is constructed by translating the English NLI datasets SNLI, MNLI, and XNLI. The training set of KorNLI is composed of the 942,854 translated examples of SNLI and MNLI by machine translation, and the validation set and the test set include 2,490 and 5,010 examples, respectively, translated from XNLI by expert human translators.

Some problems with this NLI task and its datasets may impact task performance. Problems arose from the translation process that may have impacted model performance for the NLI task since the KorNLI dataset is a translated version from the English NLI datasets.

There are awkwardly translated sentences in Korean that make it difficult to judge the relationship between sentences in pairs. In particular, the training set of the KorNLI dataset is constructed by machine translation, so such problems are more likely to arise. Examples (5) and (6) are extracted from KR-BERT’s mispredicted cases in the KorNLI dataset.

(5) **Premise:** 타이싱 컴패니에 가구와 은에는 29, 122는 도자기를 위해.
thaising khemphayniey kakwuwa uneynun 29, 122nun tocakilul wihay.
“29 are for furniture and silver in the Taixing Company and 122 are for ceramics.”

Hypothesis: 자기는 가구와 은 이상이었다.
cakinun kakwuwa un isangiessta.

“The number of porcelain was more than furniture and silver.”

Label: Entailment

In Example (5), it is hard to understand the meaning of the premise sentence. While it can be read that the number of the furniture and silver is 29 and that of ceramics is 122, and therefore that the relation to predict is ‘entailment’, Korean is not usually written in this manner. The sentence does not include the quantity units usually required in Korean as they may not have been in the English sentence, and it is not clear that those things exist in the ‘Taixing Company.’ The hypothesis sentence is also awkward in its comparison of the quantity of porcelain with those of furniture and silver.

Example (6) is a sample sentence pair which shows another problem with the translated dataset.

(6) **Premise:** 이것은 헝가리에 새로 온 사람이고, 놀고 싶다면 시내를 운전으로 시내를 벗어나야 할 것이다.

ikesun hengkaliey saylo on salamiko, nolko siphtamyen sinaylul wuncenulo sinaylul pesenaya hal kesita.

“This is a newcomer to Hungary, and if you want to play, you’ll have to drive out of town.”

Hypothesis: 마을에는 새로운 사람들이 놀 수 있는 몇 개의 장소가 있다.

mauleynun saylowun salamtuli nol swu issnun myech kayuy cangsoka issta.

“There are several places in town where new people can play.”

Label: Contradiction

In Example (6), the problem is that the Korean translation of the premise sentence is awkward. Also, models will have to know that the meaning of the token 마을 maul “village” in the hypothesis is the same as that of the token 시내 sinay “downtown” in the premise to predict the relationship of the pair as ‘contradiction.’ We assume that both tokens were written as ‘town’ in English. However, in some cases, the meaning of the Korean words 시내 sinay and 마을 maul can be contrastive, such as in downtown and residential areas. The language models may predict the relationship of the sentence pair of Example (6) as ‘contradiction’ or ‘neutral’ depending on how it modeled the two words’ meaning. Therefore, to consider the sentence relationship in a pair, the two words should have been translated into the same Korean word

when organizing the dataset.

There is also a debate about whether the original English NLI datasets themselves are appropriate to evaluate the models' ability of text understanding. The direction of NLI's determination on the authenticity of the hypothesis sentence given the premise sentence is not likely to be reflected well as originally intended. The MNLI fine-tuning code released by Google Research only classifies the entailment relations between sentences as a simple classifier without processing the two sentences' directionality. In this situation, it is essential to consider whether a 3-class prediction task, such as that of the NLI, which is more complicated than other tasks, will be performed and evaluated well.

Moreover, as Gururangan et al. (2018) and Jiang and Marneffe (2019) pointed out, the NLI dataset has the problem of predicting the label to some extent by referring only to the hypothesis sentences without considering premises. There is a tendency in the hypotheses that, if they include a negation or vagueness, they would contradict the premises. If a model has captured and learned such characteristics of only the hypothesis sentences, there is a risk that the relationship between the two sentences has not been adequately captured even if the NLI performance is high.

These problems in the related tasks should be corrected and improved in the long-term to better reflect the linguistic phenomena the task is intended to capture.

5.1.4 Semantic Textual Similarity

The datasets for the Semantic Textual Similarity (STS) task include sentence pairs annotated with a similarity score between the two sentences, ranging from 0 (completely dissimilar) to 5 (completely equivalent). The score indicates the semantic distance or similarity between the vector representations of the sentences.

We use a dataset named KorSTS, which is provided by the KakaoBrain benchmark dataset for Korean Natural Language Understanding, as in the Korean NLI task above. The training set, the validation set, and the test set are constructed through translation from an English STS dataset, STS-B (Cer et al. 2017). STS-B contains sentence pairs collected from news articles, video and image captions, and natural language inference datasets. KorSTS includes 5,749, 1,500, and 1,379 examples in the training set, the validation set, and the test set, respectively. Performance of models in this task are evaluated by Spearman’s Rank Correlation Coefficient, one of the metrics indicating the correlation between two variables.

The Spearman’s Rank Correlation Coefficient (SCC) is a nonparametric measure of the monotonicity of the relationship between two samples, commonly used as a metric for STS. Here, the two samples correspond to the vector-shaped model predictions and actual labels from the task data. It is similar to the Pearson product-moment correlation coefficient (PCC), except the input vectors are transformed to include value rankings. The SCC formula is represented in Equation (12) (Zwillinger and Kokoska 1999), measuring

the correlation between ranks. In the equation, u_i and v_i represent the rank of the i^{th} observation in the first and the second samples, respectively, and n indicates the length of the samples.

$$r_s = \frac{n \sum_{i=1}^n u_i v_i - \left(\sum_{i=1}^n u_i \right) \left(\sum_{i=1}^n v_i \right)}{\sqrt{\left[n \sum_{i=1}^n u_i^2 - \left(\sum_{i=1}^n u_i \right)^2 \right] \left[n \sum_{i=1}^n v_i^2 - \left(\sum_{i=1}^n v_i \right)^2 \right]}} \quad (12)$$

This value ranges from -1 to 1, in which +1 and -1 indicate an exact monotonic relationship, and 0 represents no correlation between the vectors. The correlation is interpreted according to its absolute value as in Table 8.

Table 8: Rule of thumb for interpreting the size of a correlation coefficient (Hinkle et al. 2003)

Size of Correlation	Interpretation
.90 to 1.00 (-.90 to 1.00)	Very high positive (negative) correlation
.70 to .90 (-.70 to -.90)	High positive (negative) correlation
.50 to .70 (-.50 to -.70)	Moderate positive (negative) correlation
.30 to .50 (-.30 to -.50)	Low positive (negative) correlation
.00 to .30 (.00 to -.30)	Negligible correlation

The KorSTS dataset includes similar problems to the KorNLI dataset since it includes the sentences translated from the English STS-B dataset. However, we suppose that STS would be less impacted by the problems affecting NLI, because STS determines the similarity between sentence embeddings rather than an elaborate semantic relationship such as entailment or contradiction.

Nevertheless, it is also important to note that translation can affect the model judgment of the similarity between sentences.

5.1.5 Sentiment Analysis

The most popular dataset for Korean Sentiment Analysis is the Naver Sentiment Movie Corpus (NSMC)³, structured around a binary classification problem of movie reviews. The NSMC dataset has two labels for prediction: positive and negative, which show the movies' stance in the review texts. The data contains 150K reviews for the training set and 50K for the test set, evaluating models by accuracy. In this dissertation, we extracted 20K reviews from the training set to utilize as the validation set, and used the remaining 130K reviews as the training set. Graduate students in our laboratory carried out the main work of splitting the data.

Additionally, we performed a Hate Speech Detection (HSD) task as part of sentiment analysis. We used the Korean Hate Speech dataset (Moon et al. 2020), which consists of Korean online entertainment news articles' comments. The dataset includes 9,381 human-labeled comments: 7,896 for the training set, 471 for the validation set, and 974 for the test set.

The dataset includes three types of annotations: the type of social bias, the existence of gender bias, and whether the comment is hate speech, which is our focus in the hate speech detection problem. There are three labels for hate speech detection: Hate, Offensive, and None.

³<https://github.com/e9t/nsmc>

- Hate: The comment displays aggressive stances towards individuals or groups.
- None: The comment does not include any aggressive stances.
- Offensive: In between the two labels, ‘Hate’ and ‘None’; the comment is not hateful or insulting but may be considered offensive due to poor taste (e.g., rude expressions and sarcasm).

We used an F1 score for the evaluation metric of this task according to the public leaderboard⁴.

5.2 Experiments

5.2.1 Experiment Details

We evaluate the performance of the models KR-BERT and KR-BERT-MEDIUM implemented in this dissertation based on the tasks described in 5.1, comparing them with those of existing Korean-based models. The baseline models include the M-BERT model of Google Research, the character-level KorBERT model of ETRI, and KoBERT of SKT Brain. Other Korean models were also released, but only those whose token units are syllable characters, not morphemes, were referred to. For M-BERT and KorBERT, we reproduced the fine-tuned task results using the provided tokenizer, vocabulary, configuration, and model checkpoints, since M-BERT has no official results on Korean tasks and KorBERT did not specify the exact datasets used for the reported model

⁴<https://www.kaggle.com/c/korean-hate-speech-detection>

performance. In the case of KoBERT, it is distributed through a PyTorch-based Transformers library, and its task performances are reported by Park (2020a) and Park (2020b), so we used them for comparison.

We used a Google Cloud TPU v3-8 or GPUs (Tesla V100 GPU, Titan XP, and a combination of Titan XP and Titan RTX) depending on the situation. In particular, the Tesla V100 GPUs are supported by the National IT Industry Promotion Agency (NIPA).

For the task datasets STS, NLI, NSMC, and HSD, we modified and utilized the TensorFlow-based BERT fine-tuning codes released by Google Research so that they can handle the corresponding Korean task datasets. For these tasks, we used the Adam optimizer, a training batch size of 128, an evaluation batch size 8, a max sequence length (maxlen) of 128, and a learning rate of $5e-5$, and fine-tuned the models for five epochs over each dataset. For HSD, we left the performance of KoBERT empty since only the F1 score of the validation set is reported for the model. The validation set performance of KoBERT is mentioned in Chapters 6 and 7 when analyzing model performances on sentiment analysis.

We assessed the NER and Question Answering (QA) tasks using the PyTorch-based fine-tuning codes released by Park (2020a) and Park (2020b). Since the codes require PyTorch-based model checkpoints, we used the released PyTorch version model of KorBERT and M-BERT and the PyTorch-converted versions of KR-BERT and KR-BERT-MEDIUM. We set the experiment parameters the same as those of Park (2020a) and Park (2020b) for direct com-

parison, using a training batch size of 32, an evaluation batch size of 64, a maxlen of 50, a learning rate of 5e-5, and 50 training epochs for NER, and a training batch size of 32, an evaluation batch size of 32, a maxlen of 512, and a learning rate of 5e-5 for question answering (KorQuAD). Here, we left the KorQuAD result of KorBERT unreported since the fine-tuning code imports the libraries relevant to the question answering task from the Transformers, which do not allow using the provided tokenizer of KorBERT.

5.2.2 Task Results

We report the model performances for the Korean NLP tasks described above in Table 9. Overall, the KR-BERT performances are high in most tasks, and the expanded model KR-BERT-MEDIUM recorded higher performances depending on the task but showed similar or slightly lower results than other models in some cases.

Table 9: The model performances for Korean NLP tasks

	NER (F1)	QA (EM / F1)	NLI (Acc)	STS (Spearman × 100)	NSMC (Acc)	HSD (F1)
M-BERT	84.20 (reported)	70.42 / 90.25 (reported)	74.17	73.42	86.82	52.03
KorBERT	87.44	n/a	78.28	77.30	89.81	54.33
KoBERT	86.11 (reported)	52.81 / 80.27 (reported)	79.62 (reported)	81.59 (reported)	89.63 (reported)	n/a
KR-BERT	87.10	73.62 / 91.15	77.13	78.70	89.74	54.53
KR-BERT -MEDIUM	86.15	52.48 / 79.49	75.29	78.34	90.29	57.91

For NER, the KorBERT performance is the highest with an F1 score of 87.44, with our KR-BERT F1 score of 87.10 following by a small difference. The F1 score of KR-BERT-MEDIUM is 86.15, higher than those of KoBERT and M-BERT. The reported performances of KoBERT and M-BERT refer to Park (2020b). For QA, KR-BERT recorded the highest EM of 73.62 and an F1 score of 91.15, followed by M-BERT. The QA result of KR-BERT-MEDIUM is lower than both while being similar to that of KoBERT. The reported performances of KoBERT and M-BERT refer to Park (2020a).

In NLI, the reported accuracy of KoBERT (Park 2020c) is the highest at 79.52, followed by KorBERT and KR-BERT, and then by KR-BERT-MEDIUM. KR-BERT showed an accuracy of 77.13, higher than KR-BERT-MEDIUM, and the KR-BERT-MEDIUM model recorded higher accuracy than that of M-BERT. In STS, the reported KoBERT performance (Park 2020c), 81.59, is the highest value of the Spearman’s Rank Correlation Coefficient (SCC). The next highest SCC record is 78.70 by KR-BERT, with KR-BERT-MEDIUM following it with 78.34. These values are higher than those of KorBERT and M-BERT.

Finally, in the two sentiment analysis tasks, NSMC and HSD, KR-BERT-MEDIUM reported the highest performances among the evaluated models. In NSMC, KR-BERT-MEDIUM showed an accuracy of 90.29, and KR-BERT showed a similar performance to those of KoBERT (Park 2020c) and KorBERT; all of them are being higher than the M-BERT result. For HSD, KR-BERT-MEDIUM recorded an F1 score of 57.91, and is followed by KR-

BERT, being similar to that of KorBERT and higher than the M-BERT performance.

Based on the results above, we obtained comparable task performances in general with the KR-BERT model, the primary model in this study. The model recorded remarkable results in NER, QA, NSMC, and HSD, and showed adequate performance on the STS and NLI tasks. The differences between KR-BERT and other Korean models, which we believe contributed to performance, are: 1) sufficient training with smaller training data; and 2) the size and composition of the vocabulary.

First, we trained KR-BERT for 2M steps based on small-scaled training data. KorBERT’s training dataset is about ten times bigger than our model. KoBERT is reported to be trained for about 20M steps with their training data, which is similar in size to ours. HanBERT was trained for 6M steps with a dataset about 20 times bigger than ours, though it is not used as a baseline model in this study. The number of steps is related to the training data size and the number of epochs, as in Equation (13), where one epoch indicates one cycle of training for the entire dataset. Accordingly, it is expected that if the training data is small, the model could perform sufficient training for the dataset with a smaller number of steps. Further, this would lessen the training time required for the model.

$$\text{number of steps} = \frac{(\text{number of epoch} \times \text{data size})}{\text{batch size}} \quad (13)$$

Despite the smaller number of steps and the smaller data size, KR-BERT recorded MLM accuracy of 77.34, which is higher than that of KoBERT, at around 75.00. Other models' training performances are unknown. Task performance, as well as training performance, do not appear to be affected by the small number of steps. KR-BERT shows better performance in many tasks than models trained using larger training datasets, and in some tasks, better than KoBERT, trained for a larger number of steps. This could be the result of the language modeling of our model being sufficient to learn linguistic knowledge well from texts.

Moreover, KR-BERT's vocabulary is similar in composition to KorBERT but smaller in size, reducing the burden of MLM computation. While our vocabulary size is slightly larger than that of KoBERT, the proportion of symbols and foreign language letters in our vocabulary is higher than KoBERT's. Such vocabulary has undergone heuristic refining processes to include more Hangul word and subword tokens, and various tokens expected to be used in various domains. Accordingly, the model could perform adequate processing for many task datasets, formal and informal, written in different styles. This is assumed to be why KR-BERT performs well for most of the NLP tasks in this study.

Moreover, our model's advantage as a monolingual Korean model compared to M-BERT is that the model only deals with Korean texts and learns Korean linguistic knowledge, not devoting time and hardware for computation for other languages. The monolingual vocabulary is also advantageous for the

model’s MLM performance since the multilingual model uses all the foreign language tokens in the vocabulary as the candidate label of masked tokens. The presence of vocabulary entries and texts of other languages may affect the multilingual model’s Korean language representation modeling, reducing performance.

KR-BERT-MEDIUM is an additional model trained as a trial with an expanded version of KR-BERT’s dataset, with a vocabulary composed by the WordPiece model without any postprocessing. This model recorded comparable performance in NER and STS, the best results in NSMC and HSD, and performed relatively lower in NLI and QA. In general, the improvement of performance from KR-BERT is not satisfactory, except for in NSMC and HSD.

The reason for such performance is that while the training data has grown in size, the number of steps is still only 2M. While we stopped training, since the model training loss seemed to converge with less improvement around the point of 2M steps, it is the same number as KR-BERT, whose training data is only about 1/4 of KR-BERT-MEDIUM. Further training for more steps may have been required, given that KR-BERT-MEDIUM’s MLM and NSP performances are much lower than KR-BERT.

Additionally, the data domains and stylistic properties that compose KR-BERT-MEDIUM’s training data may have affected its task performances. The model added legal texts and a comment dataset (accounting for 9.3GB of the total 12.37GB data) to KR-BERT’s training data, and the sequences included

in the comment dataset are likely to be one-sentence comment documents and short in length. In such cases, BERT would not have correctly performed NSP, a subtask for learning the relationship between sentences. It may have affected the performance of the model in QA, NLI, and STS, which are problems of recognizing the relationship between two sentences in a pair. While the sentence similarities in STS are relatively simple to be processed by estimating the distance between two sentence embeddings, the relationship between the sentences in the datasets for QA and NLI are defined as complicated, resulting in more challenging label prediction.

The comment sequences contain the stylistic properties of informal texts. Such texts include abbreviations, coinages, emoticons, spacing errors, and typos, unlike the sentences extracted from news texts, encyclopedias, and books. The NSMC and HSD data, along with the user-generated subjective short writings, are expected to have similar stylistic properties to the comment data. In this respect, KR-BERT-MEDIUM would have reported better results in sentiment analysis than other models.

We assume the diversity of data domains or styles to be a factor influencing our trial through the expanded model. We can not assert the effect of data diversity until it is verified by sufficient experiments designed with various data compositions. Therefore, we sought to focus more on the performance of our primary model, KR-BERT.

5.3 Chapter Summary

In this chapter, we observed the performances of Korean-specific embedding models described in Chapter 4 using several Korean NLP tasks. In 5.1, we explained the tasks reflecting varied linguistic knowledge and the details of them. The tasks tested in this dissertation include NER, QA, NLI, STS, sentiment analysis on movie reviews, and hate speech detection.

In 5.2, we provided the experiment details for the NLP tasks, including their fine-tuning codes and parameters, with a simple description of the baseline models, namely M-BERT, KorBERT, and KoBERT, and a comparison of their results. We reported our models' task performance with those of the baseline models and analyzed the factors involved in achieving such performance, considering model details and properties of training data and task datasets.

6 An Extended Study to Sentiment Analysis

We described the Korean-specific pre-trained BERT models trained in this dissertation and their task performances in Chapters 4 and 5. In this chapter, we extend our study to apply the models to sentiment analysis using additional sentiment information. While studies on sentiment analysis include several branches depending on the span and the target of sentiment expressions, we consider only sentence-level stance classification in this study. As introduced in 2.3, several studies have attempted to improve task performance on sentiment analysis using various features. We focus on the usage of sentiment lexicons among the varied sentiment features.

The content of this chapter is based on our existing work (Lee and Shin 2020).

6.1 Sentiment Features

6.1.1 Sources of Sentiment Features

A sentiment lexicon refers to a list of subjective words that have a sentiment orientation. The sentiment polarity of words is commonly annotated in many lexicons, and in some cases, additional information such as the subjectivity and intensity of words are included.

In this dissertation, we considered Korean pre-defined sentiment lexicons, namely the Korean Sentiment Analysis Corpus (KOSAC) (Shin et al. 2012) and the KNU Korean Sentiment Lexicon, as a basis for sentiment feature con-

struction. We composed sentiment features based on them and constructed the sentiment embeddings used for model training.

Korean Sentiment Analysis Corpus (KOSAC)

The KOSAC corpus contains 17,582 annotated sentiment expressions from 332 documents and 7,744 sentences from the Sejong Corpus and news articles. The sentiment expressions include annotations of expressive types, subjectivity types, sentiment targets, nested sources, polarity, intensity on phrases or words, and sentence-level subjectivity. The annotation of these items is based on morpheme units, where the word roots, the case markers, and the suffixes are separated in sequences by part-of-speech tagging.

It is similar to the MPQA corpus (Wiebe et al. 2005), which contains annotated sentiment phrases for about 10,000 sentences, and uses them as a gold standard for sentiment-related tasks. Similarly, KOSAC can also be used as data for training and verifying models of sentiment analysis.

We utilized the polarity and intensity values assigned to each token in the KOSAC word list as sentiment features among the various attributes annotated in the corpus. 16,362 morphemes are annotated with polarity and intensity values, and the polarity values include five classes: POS (positive), NEUT (neutral), NEG (negative), COMP (complex), and None (no polarity value). COMP has both positive and negative polarity values in one token, such as in the Chinese character expression ‘幸不幸,’ “happiness and unhappiness” (Shin et al. 2012). The four intensity values include High, Medium, Low, and

None (no intensity value). These values show how strong the sentiment is in the token. Tables 10 and 11, respectively, show the distributions of the polarity and intensity values in KOSAC.

Table 10: The distribution of the polarity tags in KOSAC

Tag	POS	NEUT	NEG	COMP	None	Total
frequency	7,367	1,005	6,994	439	557	16,362
(ratio)	(45.03%)	(6.14%)	(42.75%)	(2.68%)	(3.40%)	

Table 11: The distribution of the intensity tags in KOSAC

Tag	High	Medium	Low	None	Total
frequency	2,305	11,437	2,089	531	16,362
(ratio)	(14.09%)	(69.90%)	(12.77%)	(3.25%)	

KNU Korean Sentiment Lexicon

This sentiment lexicon is constructed by Kunsan National University, and includes 14,843 entries of sentiment expressions collected from several sources. It is reported that the lexicon consists of subjective words representing universal human feelings rather than those used in specific domains.

The list of positive and negative words was constructed from the following sources:

- The top 2500 positive and negative words extracted from the BiLSTM polarity classification results on the words' glosses in the Standard Ko-

rean Language Dictionary of the National Institute of the Korean Language

- The list of positive and negative words provided by Gim (2004)
- The list of positive and negative words translated from SentiWordNet 3.0 (Baccianella et al. 2010) and SenticNet-5.0 (Cambria et al. 2018)
- The list of recent popular online abbreviations, coinages, and emoticons introduced in Wikipedia

Each word in the lists above is based on character units without a refined tokenization process, and each of them is annotated with the 5-class polarity values according to the Likert scale: very positive, positive, neutral, negative, and very negative. Table 12 describes the distributions of the polarity values in the lexicon.

Table 12: The distribution of the polarity tags in KNU Sentiment Lexicon

Tag	Very Positive	Positive	Neutral	Negative	Very Negative	Total
frequency	2,597	2,266	154	5,029	4,797	14,843
(ratio)	(17.50%)	(15.27%)	(1.04%)	(33.88%)	(32.32%)	

6.1.2 Assigning Prior Sentiment Values

We considered three sentiment feature composition options based on the two sentiment lexicons described above and attempted to choose one of them to train a sentiment-combined BERT model.

- 1) Using the 3,267 single-morpheme tokens of KOSAC to compose an annotated sentiment word list
- 2) Using the 15,862 single- and multi-morpheme tokens of KOSAC to compose the sentiment word list
- 3) Using the 15,862 single- and multi-morpheme tokens of KOSAC and the 6,542 single-word entries of the KNU lexicon to compose the sentiment word list

First, if only the single-morpheme tokens of KOSAC are used, the polarity and intensity values annotated in KOSAC are assigned to the tokens. For example, a token 아름답 alumtap “beautiful” has corresponding polarity and intensity values ‘POS’ and ‘Medium’ respectively in KOSAC, so these values are assigned to the token. If a token in a BERT-based vocabulary is not included in the KOSAC word list, it is assigned the default value ‘None’ for its polarity and intensity.

On the other hand, considering the second option above, KOSAC contains many multi-morpheme tokens since the entries in KOSAC result from part-of-speech tagging such as 똑같 tokkath “same” + 은 un (an ending). Such words are concatenated to construct the complete token as 똑같은 tokkathun “same” and added to our model’s sentiment word list.

We excluded some words as an exception in cases of words including grapheme-level morphemes, as in 가능 kanung “able” + 하 ha “be” + ㄴ n (suffix), and cases in which the part-of-speech-tagged form of a word is

different from the original form, as in 행복해서 hayngpokhayse “because [I am] happy” and 행복 hayngpok “happy” + 하 ha “be” + 아서 ase (suffix). In other words, the concatenated tokens that match with a BERT vocabulary are added to our sentiment word list, with the ratio of sentiment words out of the vocabularies represented in Table 14, provided later. In the KNU sentiment lexicon case, we adjusted the polarity values and added the estimated intensity values for each token to obtain the same form of values as KOSAC, since it only includes the polarity values, which are differently defined from those of KOSAC. First, we assigned the polarity ‘POS’ to tokens with ‘very positive’ and ‘positive’ polarity values, ‘NEG’ to tokens with ‘very negative’ and ‘negative’ values, and ‘NEUT’ to tokens with ‘neutral’ polarity values in the KNU lexicon. The intensity values of ‘very positive’ and ‘very negative’ tokens in the KNU lexicon are assigned as ‘High,’ ‘positive’ and ‘negative’ tokens are assigned as ‘Medium’, and ‘neutral’ tokens are assigned as ‘Low.’

However, it is not likely that such an intensity match is very accurate. The KOSAC tokens with the polarity value ‘NEUT’ do not necessarily have low intensity, but may have various intensity values. However, it would be best to perform such matching of intensity values, since, from an intuitive perspective, the strength of the subjectivity of neutral tokens is expected to be lower than those with more evident polarity values such as ‘POS’ and ‘NEG’. Moreover, these prior sentiment values are only the initial values and will be modified and updated based on contextual information through model training. Table 13 represents the alignment of polarity and intensity values

between the two sentiment lexicons above.

Table 13: The alignment of sentiment values in KOSAC and the KNU lexicon

KNU Tag	Very Positive	Positive	Neutral	Negative	Very Negative
KOSAC Polarity	POS	POS	NEUT	NEG	NEG
KOSAC Intensity	High	Medium	Low	Medium	High

If multi-morpheme tokens in KOSAC and the KNU lexicon entries are added to our sentiment word list as in the third option above, a variety of word forms in Korean can be captured and assigned the proper sentiment values. For example, assume that the shortest word root form 좋 coh “good” is assigned prior sentiment values but not the varied word forms such as 좋은 cohun “good,” 좋아서 cohase “because [it’s] good,” and 좋지만 cohciman “[It’s] good, but. . .” are not included in our sentiment word list. The varied word forms do not have any pre-defined sentiment values; thus, it will be hard to perform proper sentiment processing for those tokens and the text sequences that include the tokens.

However, the multi-morpheme tokens in KOSAC and the words in the KOSAC lexicon include the varied word forms described above so that the prior sentiment values defined in those sources can be assigned to the tokens. Additionally, the KNU lexicon contains the sentiment values for online abbreviations, coinages, and emoticons, not included in KOSAC. Such expressions will help sentiment processing of texts using such words if they are also included in BERT vocabularies.

These differences can be visualized, as in Figure 9. A sentiment word list

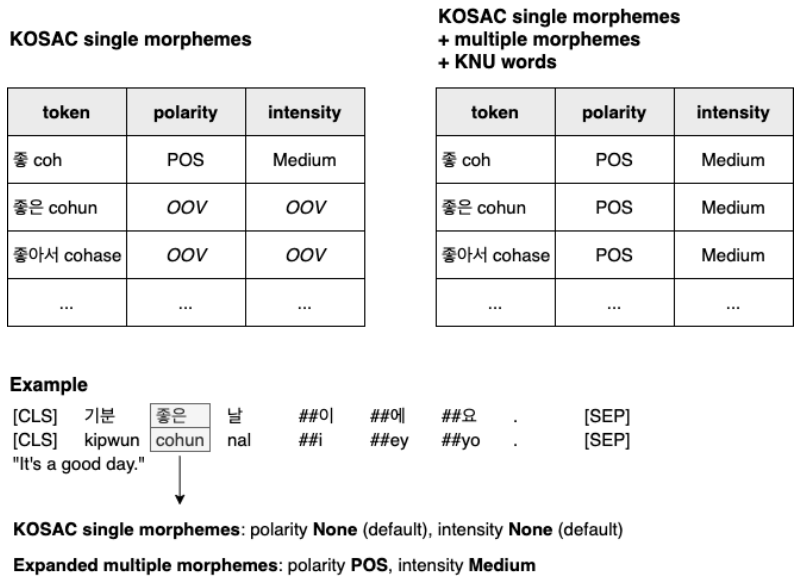


Figure 9: Comparison of sentiment word matching depending on the lexicon

consisting of only single-morpheme tokens assign sentiment values only to the single-morpheme token 좋 coh “good” as in the left of the figure, assigning the default values ‘None’ to other out-of-vocabulary (OOV) words. However, the expanded multi-morpheme sentiment word list on the right of the figure includes the variation forms 좋 coh “good,” 좋은 cohun “good,” and 좋아서 cohase “because [it’s] good,” assigning prior polarity and intensity values to them.

More matching of sentiment words in BERT vocabularies and training data would result in a better sentiment understanding and processing of models based on more accurate sentiment values. Table 14 shows the ratio of sentiment words constructed based on the three options above out of the 16,424

tokens in the KR-BERT vocabulary, which is the basic model in this dissertation.

Table 14: The ratio of sentiment words in the KR-BERT vocabulary

Sentiment word composition	KOSAC single morphemes	KOSAC single + multi morphemes	all KOSAC morphemes + KNU words
Ratio of sentiment words	2,062 (12.55%)	2,522 (15.36%)	2,634 (16.04%)

If we use only the single-morpheme tokens in KOSAC, 2,062 of 16,424 tokens in the KR-BERT vocabulary are assigned prior sentiment features. 460 tokens are added to this if both of the single-morpheme and multi-morpheme tokens in KOSAC are used, being matched with 2,522 tokens in the vocabulary. The last option adds 112 more sentiment words to the list, and 2,634 tokens in the vocabulary are assigned polarity and intensity values. Here, the number of sentiment tokens added by the use of KNU entries is relatively small. This indicates that the sentiment words included in the multi-morpheme KOSAC tokens and the KNU lexicon overlap significantly, resulting in a smaller number of newly-added tokens in our sentiment word list.

We compared the three options of composing our sentiment word list to decide the range of sentiment words used for sentiment value assignment and processing in this dissertation. For verification, we compared the ratio of sentiment words in the training data of KR-BERT.

Table 15: The ratio of sentiment words in the KR-BERT training data

Sentiment word composition	KOSAC single morphemes	KOSAC single + multi morphemes	all KOSAC morphemes + KNU words
Ratio of sentiment words	44.20%	49.71%	50.21%

Table 15 shows the ratio of sentiment words in KR-BERT’s training data averaged for all the sentences, depending on the lists by adding tokens from the two Korean sentiment lexicons incrementally. As the ratio of sentiment words grows, more tokens in the training data are assigned the prior sentiment values from the lexicons. The model using more sentiment words obtains more initial token-wise sentiment values expected to help sentiment analysis of sentences or documents.

In the table, the ratio of sentiment words in the training data dramatically increases when the KOSAC multi-morpheme tokens are added to the sentiment word list. However, adding the KNU words to the list does not affect the ratio much. As also described in Table 14, the sentiment words included in the multi-morpheme KOSAC tokens and the KNU lexicon overlap significantly, resulting in a smaller number of new tokens added to our sentiment word list. Additional adjustment of polarity and intensity values is required when using the KNU entries, which is not necessarily to be pursued, considering an immaterial increase in the ratio of the sentiment words in the dataset. Therefore, we did not include tokens from the KNU lexicon in our sentiment word list in this study.

On the other hand, only 15.36% of the KR-BERT vocabulary tokens and 49.71% of tokens in its training data are assigned prior sentiment values from KOSAC, even if the multi-morpheme tokens are added. So we considered a method to increase these ratios by applying additional processes to assign sentiment values to the unmatched tokens, as follows:

- Existing sentiment words maintain their sentiment values.
- BERT special tokens such as [SEP], [CLS], [PAD], [UNK], and [MASK] are assigned the polarity and intensity values ‘None’ as default.
- Monosyllabic tokens not included in KOSAC are assigned the values ‘None’ for polarity and intensity.
- For the tokens composed of more than two syllables:
 - If they are proper nouns, the sentiment values ‘None’ are assigned to them.
 - If they are not proper nouns and consist of Hangul characters, we apply the Minimum Edit Distance (MED) algorithm based on a Levenshtein distance, assigning the closest token’s sentiment values. For each target token, the tokens that start with the same syllable are preferred among the candidate close to the tokens to process the varied forms from the same root of words.

This process assigns polarity and intensity values to all the tokens in the vocabulary and the training data, showing a ratio of sentiment words of 100%,

both in the vocabulary and the corpus.

For further comparison, we trained the BERT models based on the KR-BERT vocabulary and training data using the matched sentiment words constructed by using:

- 1) Only single-morpheme tokens in KOSAC
- 2) All the single- and multi-morpheme tokens in KOSAC
- 3) All of the vocabulary tokens processed with the MED method

All three models are trained for 100,000 steps with the Adam optimizer, a max sequence length of 512, a training batch size of 64, and a learning rate of 1e-4, and up to 77 tokens in each sequence are masked for each model. Table 16 compares the training performances of the models.

Table 16: Training performances of models with different sentiment word compositions

Sentiment word composition	single-morpheme KOSAC words	KOSAC single +multiple morphemes	all the tokens in the vocabulary
loss	1.2025661	1.5244046	1.3857222
MLM Acc	71.99	68.53	71.30
MLM Loss	1.2265519	1.4288566	1.2460587
NSP Acc	97.50	95.38	95.75
NSP Loss	0.055956222	0.1172123	0.09755086

In the table, the model using the single-morpheme sentiment words in KOSAC reported the best training performances, while the three models do not show remarkable differences from each other. Theoretically, the training

performance should grow with more sentiment tokens matched in the vocabulary and training data, while the actual performance shows the opposite. Furthermore, training for more steps can lead to this difference increase.

It is only possible to compare such sentiment word composition options in indirect methods via training task performances, not revealing specific factors that affected the results. However, we suppose that the word forms that concatenated multiple morphemes or matched with other word forms by MED were not frequent enough in the training data to help the language modeling of our BERT-based model. Also, the proper nouns that we assigned 'None' for polarity and intensity values might not help the model learn linguistic knowledge; the name of a specific person or organization may have other sentiment values. The additional works we considered to assign the sentiment values to more tokens in the BERT vocabulary did not significantly affect the model's training performances.

Additionally, considering performance gain, using only single-morpheme tokens in KOSAC as sentiment words is the simplest and most cost effective method. Therefore, we decided to construct sentiment features based on the prior polarity and intensity values of single-morpheme tokens included in KOSAC.

6.2 Composition of Sentiment Embeddings

First of all, we refer to each token's sentiment values in the vocabulary we used to train our BERT-based sentiment-combined model. Example (7) is part

of an excerpt from the Korean movie review dataset NSMC.

(7) 유치하고 완전 최악이었음.

yuchihako wancen choyakiessum.

“It was childish and totally terrible.”

Table 17 shows the tokens composing the sentence of Example (7) processed by the WordPiece tokenizer and with corresponding KOSAC polarity and intensity values for each token.

Table 17: An example of tokens and assigned sentiment values

	[CLS]	유치	##하고	완전	최악	##이	##있	##음	.	[SEP]	[PAD]	[PAD]
	[CLS]	yuchi	##hako	wancen	choyak	##i	##ess	##um	.	[SEP]	[PAD]	[PAD]
	[CLS]	“childish”	“and”	“totally”	“terrible”	“be”	“[past]”	“[noun suffix]”	“.”	[SEP]	[PAD]	[PAD]
polarity	None	None	None	NEG	NEG	POS	NEG	NEG	None	None	None	None
intensity	None	None	None	Medium	High	Medium	Medium	Medium	None	None	None	None

Polarity and intensity information is encoded in the form of an integer vector composed of indices, whose indices come from the polarity vocabulary and the intensity vocabulary. For example, assume that we have 5 types of polarities, ‘None, POS, NEUT, NEG and COMP’, and this list can be converted into a list of indices: ‘0, 1, 2, 3, 4.’ If a token in a sequence has a polarity value of ‘POS,’ then ‘1’ is assigned to that token, and it becomes an element of the polarity embedding corresponding to that sequence. The intensity values are also converted into an integer vector in the same manner.

The integer vectors of polarity and intensity values construct the polarity and intensity embeddings, respectively, in the BERT architecture. The polarity

and intensity embeddings are combined with existing token-based input embeddings from the BERT model, composed of the sum of the token, position, and segment embeddings by element-wise summation, as shown in Figure 10.

Input	[CLS]	이 i "this"	[MASK]	너무 nemwu "too"	재미 caymi "amus"	##있다 ##issta "##ing"	.	[SEP]	참 cam "very"	좋았다 cohassta "was good"	.	[SEP]
Token Embeddings	$E_{[CLS]}$	E_i	$E_{[MASK]}$	E_{nemwu}	E_{caymi}	$E_{##issta}$	$E_{.}$	$E_{[SEP]}$	E_{cam}	$E_{cohassta}$	$E_{.}$	$E_{[SEP]}$
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}
Polarity Embeddings	E_{None}	E_{POS}	E_{None}	E_{NEG}	E_{POS}	E_{None}	E_{None}	E_{None}	E_{POS}	E_{None}	E_{None}	E_{None}
Intensity Embeddings	E_{None}	E_{Medium}	E_{None}	E_{Medium}	E_{Medium}	E_{None}	E_{None}	E_{None}	E_{High}	E_{None}	E_{None}	E_{None}

Figure 10: Visualization of the embedding composition

The figure shows a sequence composed of the two sentences in Example (8). The WordPiece tokenizer tokenizes the sequence as in the input row on the top of Figure 10. In this case, if the token 영화 yenghwa “movie” is masked for the Masked Language Modeling (MLM) task during training, the position is filled with the token ‘[MASK]’ and the corresponding embeddings as in the figure.

(8) 이 영화 너무 재미있다. 참 좋았다.

i yenghwa nemwu caymiissta. cam cohassta.

“This movie is hilarious. [I] liked it so much.”

Since we infused sentiment information to existing token embeddings, which include position and segment information before training, we expect

our models to learn and update the pre-assigned polarity and intensity values of the tokens in the sequence according to context information by language modeling.

Token Masking and Sentiment Masking

Constructing the embeddings and the BERT-based models based on them, we do not mask polarity and intensity features but instead allow them to function as additional clues when performing MLM to train the model. They affect calculating attention weights as sentiment-related tokens would get relatively higher attention weights in the sequence.

Ke et al. (2020) included the prediction of sentiment word polarity while not masking it during the language modeling process of their SentiLARE models, containing the polarity prediction loss in its training performances. Referring to this study, we considered some cases of masking and predicting polarity and intensity, to determine which results in the best training model:

- 1) Masking and predicting each of the polarity and intensity values in the masked position
- 2) Predicting but not masking the polarity and intensity values as in Ke et al. (2020)
- 3) Neither masking nor predicting the polarity and intensity values

For the first and the second cases, the losses for token, polarity, and intensity predictions are calculated separately and then linearly combined, as in

Equation (14), to compose the MLM loss.

$$loss_{MLM} = loss_{token} + loss_{polarity} + loss_{intensity} \quad (14)$$

We calculated the losses for polarity and intensity in the same way as the MLM loss described in Equation (6) in Chapter 3. We changed the candidate tokens for MLM to the candidate polarity and intensity values, respectively. For polarity prediction, K , the number of labels is set to five, while for intensity prediction, it is set to four. M and N represent the number of masked tokens and the number of sequences in the data, respectively. $x_{m,n}$ represents the input embeddings for the m^{th} masked token in the n^{th} sequence in the training data. $y_{m,n}$ indicates the target label for the m^{th} masked token in the n^{th} sequence in the training data.

$$-\frac{1}{MN} \sum_{n=1}^N \sum_{k=1}^K \sum_{m=1}^M \left(y_{m,n}^k \times \log(\text{softmax}(x_{m,n}, k)) \right) \quad (6)$$

In the third case, at the masked position, only the token is masked and predicted using embeddings that include polarity and intensity information. So as Equation (15) shows, the MLM loss is defined by only the token prediction loss, as of that of the original BERT model.

$$loss_{MLM} = loss_{token} \quad (15)$$

Table 18 compares the performances obtained by training the sentiment-combined models for 100,000 steps based on the three cases of masking and predicting the polarity and intensity values. The models are trained using the

same training data and vocabulary as KR-BERT, all with the same sentiment features, based on KOSAC, and with the same parameters, including the usage of the Adam optimizer, a maxlen of 512, a training batch size of 64, and a learning rate of 1e-4. All the models are based on the BERT-base model scale, which consists of 12 encoder layers, 12 attention heads, and a hidden size of 768, with each of them requiring about 5 hours to train using a Google Cloud TPU v3-8.

Table 18: Training performances of the models predicting and masking tokens and sentiment values

	Mask and predict tokens and sentiments	Mask only tokens and predict tokens and sentiments	Mask and predict only tokens
loss	3.3182616	1.2344264	1.2025661
MLM token acc	54.27	71.56	71.99
MLM polarity acc	79.17	100.00	n/a
MLM intensity acc	80.92	100.00	n/a
MLM token loss	2.37334	1.2506855	1.2265519
MLM polarity loss	0.50147086	5.346803E-05	n/a
MLM intensity loss	0.4530841	4.3719585E-05	n/a
NSP acc	95.00	97.50	97.50
NSP loss	0.12870333	0.05544886	0.055956222

The loss in the first row indicates the sum of the MLM losses and NSP loss for each model in the table. The rows for MLM token acc and loss include the performances for masking and predicting tokens as in the original BERT models. The rows for MLM polarity acc and loss and MLM intensity acc and loss indicate the performances for predicting polarity and intensity, which are

not defined in the third case.

As the table shows, while NSP performances were similar, MLM performances differed between the models. The first and third cases, not masking polarity and intensity, reported better training performance. We expect that the polarity and intensity embeddings will help predict the masked tokens as additional information for language modeling when they are not masked in that position.

The MLM performances in the first and third cases are similar, while the prediction accuracy for polarity and intensity is 100%. The perfect prediction results from the small numbers of polarity and intensity candidates (only 5 and 4, respectively) not impacting model capacity. Therefore, we adopted the third case for our model training since the first case requires fewer prediction computations, not affecting performance much.

6.3 Training the Sentiment-Combined Model

We trained a sentiment-combined model using the input embedding constructed in Figure 10 based on the third case discussed in 6.2. The sentiment features of the model include single-morpheme tokens and their polarity, and intensity annotations in KOSAC. This model, named KR-BERT-KOSAC, is available on GitHub so that public access and application are allowed.

Its training details are the same as those of KR-BERT. We used the same training data and vocabulary as KR-BERT to train KR-BERT-KOSAC and trained the model for 2M steps using the Adam optimizer, a maxlen of 512,

a training batch size of 64, and a learning rate of $1e-4$. The training process required about 92 hours using a Google Cloud TPU v3-8.

Figure 11 illustrates the model training in the form of training loss, which converges to a certain level to produce stable performances before reaching 2M steps. In the figure, the x-axis represents the number of steps, and the y-axis indicates the training loss.

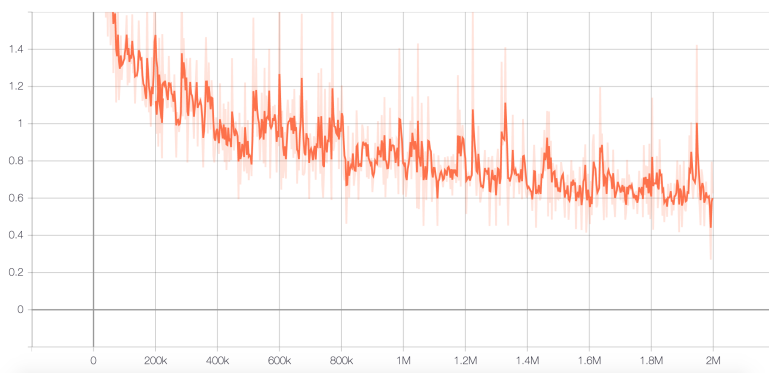


Figure 11: Training loss of KR-BERT-KOSAC

Table 19: Training performances of KR-BERT and KR-BERT-KOSAC

	KR-BERT	KR-BERT-KOSAC
global steps	2M	2M
loss	0.9154538	0.54173344
MLM acc	77.34	85.15
MLM loss	0.9511529	0.56670845
NSP acc	99.50	99.38
NSP loss	0.013327285	0.02148062

Table 19 reports the training performance of KR-BERT-KOSAC compared to that of KR-BERT. The row ‘loss’ indicates the global loss combining MLM and NSP loss, while the next four rows show the detailed performances of the two BERT training tasks. In KR-BERT-KOSAC, where the sentiment features are added to the KR-BERT architecture, MLM performance greatly improved, and the global training loss decreased even when NSP performance slightly dropped. Accordingly, we suppose that the prior polarity and intensity values referring to the Korean sentiment lexicon helped language modeling during model training, functioning as supplementary information.

This model is released on GitHub¹ and can be publicly accessed.

The effect of adding sentiment features to BERT input embeddings can be observed through sentiment analysis tasks. We fine-tuned and tested our sentiment-combined model with the sentiment analysis tasks based on the two datasets for movie review classification (NSMC) and the hate speech detection (HSD) used in Chapter 5. The resulting accuracies and F1 scores of task predictions are reported in Table 20.

For the task datasets NSMC and HSD, we modified and utilized the TensorFlow-based BERT fine-tuning codes released by Google Research. The experiment settings used for these tasks are the same as those we used in Chapter 5. For both tasks, we used the Adam optimizer, a training batch size of 128, an evaluation batch size 8, a max sequence length (maxlen) of 128, and a learning rate of $5e-5$, and fine-tuned the models for five epochs over each dataset.

¹<https://github.com/snunlp/KR-BERT-KOSAC>

Table 20: Model performances on sentiment analysis tasks

	NSMC		HSD	
	dev acc	test acc	dev F1	test F1
M-BERT	87.08	86.82	73.39	52.03
KorBERT	90.48	89.81	78.74	54.33
KoBERT	n/a	89.63 (reported)	66.21 (reported)	n/a
Park et al. (2019)	n/a	89.82 (reported)	n/a	n/a
KR-BERT	89.86	89.74	78.18	54.53
KR-BERT-KOSAC	90.30	89.82	80.00	53.81

We compared the task performances of KR-BERT-KOSAC with those of existing models, including KR-BERT. The M-BERT and KorBERT performances are our reproduction results using provided vocabularies, tokenizers, and model checkpoints. The results of KoBERT and Park et al. (2019) are reported in each source.

Park et al. (2019) is a model that added a self-attention mechanism to ELMo, recording an accuracy of 89.82 for the test set of NSMC, which was the highest at that time compared to the traditional models, including CNN and BiLSTM. KoBERT’s performance is provided as a baseline for testing other models in Park (2020c), reporting an accuracy of 89.63 for the test set of NSMC and an F1 score of 66.21 for HSD’s validation set.

KR-BERT-KOSAC tends to improve its performance in sentiment analysis problems compared to KR-BERT, a model that, while maintaining all other conditions, does not incorporate sentiment features. The accuracies of KR-BERT-KOSAC for both the validation set and the test set of NSMC increased,

recording a validation accuracy of 90.30. The F1 score for HSD’s validation set improved to 80.00 from 78.18 of KR-BERT, while that of the test set of HSD decreased slightly below KR-BERT.

These values are higher than M-BERT’s performance and are higher and lower than the KorBERT results depending on the task and data split, summarized as similar performances. While it is hard to compare the models in various perspectives since Park et al. (2019) reported only the NSMC test set accuracy, our KR-BERT-KOSAC recorded the same accuracy of 89.82 with them for the same test set.

However, this increase in performance is not satisfactory, even with a slight drop in some cases, considering the time and hardware costs required to train KR-BERT-KOSAC from scratch. Thus, we pursued a different methodology, described in Chapter 7, to further improve the performance of sentiment analysis problems using the sentiment features but with less cost.

6.4 Effect of Sentiment Features

This section attempted to check the effect of adding sentiment features to an existing BERT model, mainly when processing the sentences with their sentiment orientations from sentiment tasks.

We trained the ablated models by excluding each of the polarity and intensity features from the KR-BERT-KOSAC embeddings to evaluate the effect of the two prior sentiment values we obtained from a sentiment lexicon. The KR-BERT model corresponds to the basic model not using either the polar-

ity or intensity embeddings. Keeping the parameters and experiment settings of the sentiment-combined model unchanged (using the Adam optimizer, a maxlen of 512, a training batch size of 64, and a learning rate of 1e-4), we reconstructed the input embeddings including different feature combinations: the basic BERT embeddings combined with polarity embeddings and with intensity embeddings.

We trained all the models for 100,000 steps. First, the models show differences in language modeling ability during training, as in Table 21.

Table 21: Training performances of ablated models

	original input (KR-BERT)	+ polarity	+ intensity	+ polarity + intensity (KR-BERT-KOSAC)
global steps	100K	100K	100K	100K
loss	1.7887475	1.239531	1.3644867	1.2025661
MLM Acc	62.69	71.13	70.05	71.99
MLM Loss	1.8145926	1.2727814	1.3349932	1.2265519
NSP Acc	98.00	97.88	98.25	97.50
NSP Loss	0.049355935	0.0560604	0.042528342	0.055956222

In Table 21, the first column shows the training performance of the KR-BERT model that does not include any sentiment features in its input embeddings. The next three columns show the performances of the models that added polarity features, intensity features, and both polarity and intensity features to the basic input embeddings of KR-BERT. The last model, including both polarity and intensity embeddings, corresponds to the KR-BERT-KOSAC model described in 6.3.

According to the results, when both of the sentiment features are used in the model’s input embedding, MLM shows the best performance. Accuracy drops, and the loss increases when sentiment features are deleted from the model’s input embeddings, showing the worst performance when both of the sentiment features are removed. These results prove that the sentiment features we used in this study help train our models via language modeling.

Although the NSP performance of KR-BERT-KOSAC was slightly lower than those of other models, this difference was smaller than the increase in MLM performance. Therefore, we decided to use both the polarity and intensity features according to the advantage during training the models via language modeling.

Table 22: Task performances of ablated models

	NSMC		HSD	
	dev acc	test acc	dev F1	test F1
KR-BERT	89.86	89.74	78.18	54.53
+ polarity	89.61	89.34	77.60	54.46
+ intensity	89.85	89.37	78.00	55.84
+ polarity + intensity (KR-BERT-KOSAC)	90.30	89.82	80.00	53.81

We also compared the performance of the ablated models above in sentiment analysis tasks. The ablated models combining each of the polarity and intensity embeddings to the basic BERT input embedding are trained for 2M steps as in KR-BERT and KR-BERT-KOSAC. Table 22 shows the accuracies

on the movie review classification and the F1 scores on the hate speech detection using the same datasets and experiment settings, as described in 5.2.2.

The first row in Table 22 represents the performance of KR-BERT as the basic model, not including any of the sentiment features from the KOSAC lexicon. The next two rows show the performances of the ablated models trained using only one of either the polarity or intensity features. The last row includes the performances of KR-BERT-KOSAC from 6.3 for comparison.

First of all, KR-BERT-KOSAC reported higher performances than KR-BERT in general, except for in HSD’s test set, as observed in 6.3. Its ablated model, including only the polarity embeddings and not the intensity embeddings, showed a small decrease in performance, one also lower than that of KR-BERT-KOSAC. While it is not a significant drop, the model reported NSMC validation set and test set accuracies of 89.61 and 89.34, and HSD F1 scores of 77.60 and 54.46, respectively.

In the case of the ablated model adding only the intensity embeddings to original BERT embeddings, the validation set and test set accuracy recorded 89.85 and 89.37 in NSMC, a slight decrease from those of KR-BERT. In HSD, the model reported a validation set F1 score 78.00, a small drop from KR-BERT, and a somewhat more apparent decrease from KR-BERT-KOSAC. The HSD test set F1 score showed 55.84, increased from KR-BERT, and even higher than that of KR-BERT-KOSAC.

Initially, we expected that there would be some improvement in performance, although not as much as KR-BERT-KOSAC, with the ablated models

using only one of either the polarity or intensity features. However, the models with only one of each of the features showed a slight decrease in performance, lower than KR-BERT-KOSAC, which includes both sentiment features. This comparison of task performance is an indirect method, so it is not easy to directly identify how the two sentiment features work when infused into BERT embeddings. Nevertheless, we can conclude that the polarity and intensity features interact to improve the model performance when used together.

Additionally, we tried a simple model comparison between KR-BERT and the sentiment-combined model KR-BERT-KOSAC using attention weights. The attention weights are calculated during BERT training based on a self-attention mechanism and indicate the relative importance of tokens in a sequence.

Referring to Kovaleva et al. (2019), we visualized attention weights for each sentence in the form of 2-dimensional attention maps. The attention weights are from the final layer of each BERT-based model (KR-BERT and KR-BERT-KOSAC), and we averaged the 12 attention weight matrices from all 12 attention heads together to get one attention map for one sentence.

Kovaleva et al. (2019) supposed that there would be self-attention patterns that emphasize the effects of specific linguistic knowledge, and attempted to verify the patterns of attention weights assigned to specific types of tokens such as nouns, verbs, subjects, objects, and negative words. They predicted that the attention maps would show vertical stripes on the specific tokens related to linguistic features, based on the tokens' high attention weights.

Referring to their approach, we also compared the models, one including sentiment features and the other not containing them, by visualizing the attention weights on the sentiment words in sequences, including sentence-level sentiment stances. Examples (9) and (10) are sentences exemplifying the change of attention weights extracted from the NSMC dataset, one positive and the other negative.

(9) 이거 완전 재밋다고 하더니 한번 보니 재밋군요.

ike wancen caymisstako hatentey hanpen poni caymissskwunyo.

“I heard it’s really amusing, and now that I’ve seen it, [I can say that] it is fun.”

Polarity Label: Positive

(10) 오글거리고 유치하고 완전 최악이었음...

okulkeliko yuchihako wancen choyakiessum...

“It was cheesy and childish, and really terrible...”

Polarity Label: Negative

In Figures 12 and 13, both the x-axis and the y-axis represent the sequence of tokens for each sentence, and the darker cells show higher attention weights between the tokens. Figures 12a and 13a represent the attention maps of two example sentences using the KR-BERT model, which does not include sentiment features.

First, Figures 12a and 13a show focused attention weights at the final token [SEP], which indicates the boundary of sentences. These focused weights

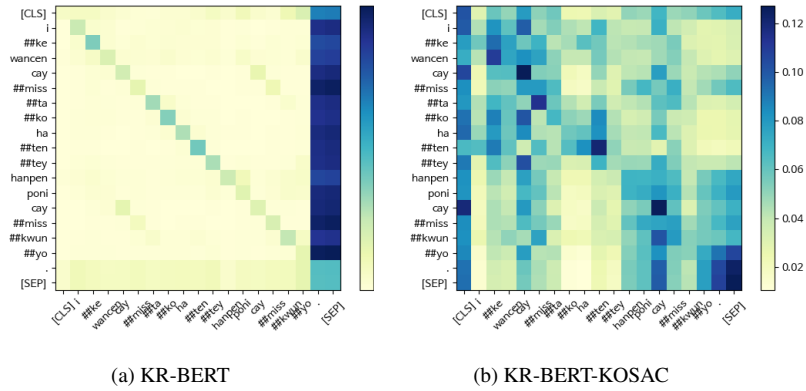


Figure 12: Attention maps for Example (9)

on the BERT special tokens such as [SEP] are similar to the vertical patterns of the attention weights commonly observed and reported by Kovaleva et al. (2019) and other existing works (Clark, Khandelwal, et al. 2019; Jawahar et al. 2019). The vertical pattern indicates the darker cells assigned to a specific token are arranged vertically in the attention map, assigning higher degrees of attention from other tokens in the sequence to the token. Additionally, they show relatively high attention weights between neighboring pairs of tokens by the diagonal line from top left to bottom right in the attention maps, as in Figures 12a and 13a. This diagonal pattern is also commonly reported in existing works.

On the other hand, the attention maps obtained by the sentiment-combined model show different patterns. In Figure 12b, attention weights are relatively spread, and some specific tokens show higher attention weights. In Example (9), the word 재밌- caymiss- “fun”, used twice, has an explicit polarity value

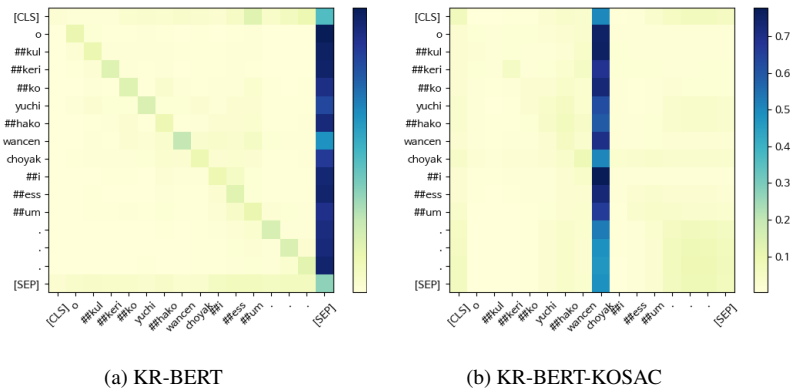


Figure 13: Attention map for Example (10)

of positive and would have an essential role in classifying the sentence to be a positive review. It seems that the attention weights around the token are high.

Additionally, Figure 13b also represents the attention maps obtained by sentiment-infused models. It is a rather extreme example, representing the pattern of attention weights in Example (10). In the sentence, a token 최악 choyak “terrible” has a polarity value ‘NEG’ and an intensity value ‘High,’ which means that the token shows extreme negativity in its meaning. So the attention weights assigned to the token are high in this case compared to those of Figure 13a.

Therefore, we speculated that sentiment-combined models would compute the attention weights of sentiment words higher in a sequence when sentiment features are infused into the models. It would help the models perform better in sentiment analysis tasks.

6.5 Chapter Summary

In this chapter, we infused additional sentiment features to the Korean pre-trained model we described in Chapters 4 and 5, KR-BERT, to improve the model’s performance in sentiment analysis. First, we constructed sentiment features based on the released Korean sentiment lexicons. We analyzed the Korean Sentiment Analysis Corpus (KOSAC) and the KNU Korean Sentiment Lexicon and decided to use the polarity and intensity values defined for the single-morpheme tokens in KOSAC as sentiment features.

In 6.2, we vectorized the tokens’ sentiment values in each input sequence matched with the sentiment lexicon, and composed the polarity and intensity embeddings. The two sentiment embeddings are then infused into the BERT original input embeddings by element-wise summation.

In 6.3, we introduced a sentiment-combined model, KR-BERT-KOSAC, that is trained based on the sentiment-infused embeddings. We explained the details of the model and training environments and compared the model performances with other existing models without sentiment features in two sentiment analysis tasks, such as movie review classification and hate speech detection. The performances of KR-BERT-KOSAC are higher than those of its baseline model, KR-BERT, which does not include sentiment features in general. However, the performance improvement is not enough concerning the effort and cost of constructing the sentiment embeddings and training the model. Therefore, it is necessary to pursue an inexpensive method with better

performances to utilize sentiment features with BERT.

Finally, in 6.4, we attempted to analyze the effects of sentiment features by comparing the performances of KR-BERT and KR-BERT-KOSAC and visualizing the attention weights of tokens in a sequence computed based on a multi-head self-attention mechanism. We also performed an ablation study by training two separate models infusing each of the sentiment features polarity and intensity used in KR-BERT-KOSAC. In conclusion, we observed that the models using sentiment features perform better than the models not using them, and utilizing both polarity and intensity features is better than including only one of them in the BERT embeddings.

7 Combining Two BERT Models

In this chapter, we attempted to improve our model performances in sentiment analysis by applying another method of infusing sentiment features into BERT. The contents of this chapter is based on our existing work (Lee and Shin 2020).

7.1 External Fusing Method

As a novel method, we provide an External Fusing method that imports different pre-trained BERT-based models separately and combines them to be fine-tuned and to produce a task result at a time, as visualized in Figure 14.

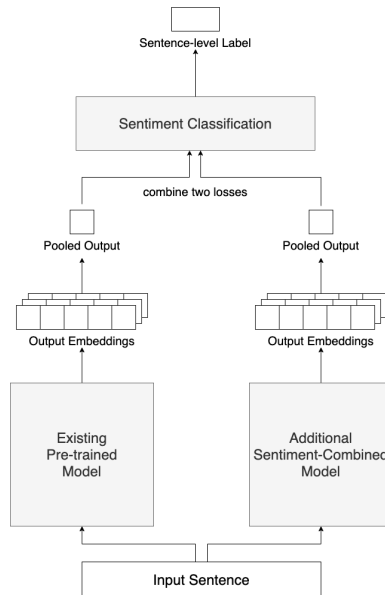


Figure 14: The structure of our external fusing method

For the pre-trained BERT models, we use M-BERT, KorBERT, and the models we described in Chapters 4 and 5, such as KR-BERT and KR-BERT-MEDIUM. For the additional sentiment-combined model on the right of the figure, we trained new models, including models using the sentiment features described in Chapter 6. Using these models, we load each model’s pre-trained weights from its checkpoint file at the fine-tuning phase for each downstream task. The loaded models then draw out the output embeddings, the final hidden states, for the input sentences from the task dataset. Each model’s output embeddings are pooled to the proper form for the task classifier layer, and each of them then computes the task loss. The two task losses are then linearly combined as in Equation (16) to be optimized at a time. This combination of losses is similar to that of combining MLM loss and NLP loss to compose a global loss during BERT training.

$$loss_{task} = loss_{pre-trained\ model} + loss_{sentiment-combined\ model} \quad (16)$$

The combined task loss is then used for the label prediction of the models for the task. Although this method requires two different pre-trained models, the method of updating model weights by referring to the task loss remains the same as the fine-tuning process of original BERT models.

The two models with different properties are expected to improve the model performance on tasks by interacting with each other through this fusing method. In particular, the existing pre-trained models report excellent train-

ing and test performances in various NLP tasks and will function well as language models for general purpose use, since they have been trained using large-scale corpora for many steps when performing language modeling. The sentiment-combined models work by being specialized for sentiment analysis, showing better performance in relevant tasks, as they include sentiment features in their embeddings. As these two models have advantages from different perspectives, we attempted to combine them to produce a better overall model.

For the general pre-trained models, we used the KR-BERT and KR-BERT-MEDIUM models trained in this dissertation, and M-BERT and KorBERT were also considered to verify the performance increases when our external fusing method was applied. These models are expected to have done enough language modeling on a large dataset to learn context information from text. While various pre-trained models work well, only a few can deal with Korean texts, and are implemented based on TensorFlow. Therefore, the models available to be directly compared with our models are the four listed above. The performances of externally combined models will be compared with those of the four pre-trained models, since we have reported the models' task performances in sentiment analysis in Chapter 5.

In this dissertation, we trained two sentiment-combined models, KR-BERT-KOSAC-SMALL and CH-BERT-KOSAC-SMALL, using different training data and vocabularies constructed from the data using the WordPiece tokenizer.

KR-BERT-KOSAC-small

The training settings of KR-BERT-KOSAC-small are the same as KR-BERT-KOSAC, except for the model scale, as this model is based on a smaller scale than KR-BERT and KR-BERT-KOSAC. Accordingly, we named the model with the suffix ‘small.’ The detailed explanation of the model scale is provided later in this chapter. We used the same 2.47GB training data as KR-BERT, including Wikipedia texts and news articles, and the same vocabulary including 16,424 tokens trained and postprocessed based on the data.

We added sentiment features to the input embedding constructed based on single-morpheme tokens annotated in the KOSAC lexicon, as in KR-BERT-KOSAC. We trained the KR-BERT-KOSAC-small model using an Adam optimizer, a max sequence length of 128, a training batch size of 64, and a learning rate of $1e-4$, for 500K steps with two Tesla V100 GPUs for 70 hours.

Table 23: Training performances of KR-BERT-KOSAC-small

	KR-BERT-KOSAC-small
global steps	500K
loss	1.532343
MLM acc	67.63
MLM loss	1.447246
NSP acc	96.38
NSP loss	0.08696271

Table 23 reports the training performance of KR-BERT-KOSAC-small. The row ‘loss’ indicates the global loss combining the MLM loss and the

NSP loss, while the next four rows show the detailed performance of the two BERT training tasks. Additionally, Figure 15 visualizes the model’s training losses up to 500K steps. In the figure, the x-axis represents the number of steps, and the y-axis indicates the training loss.

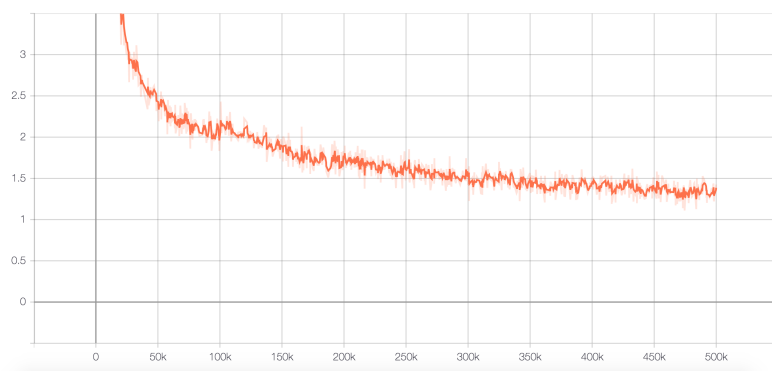


Figure 15: Training loss of KR-BERT-KOSAC-small

CH-BERT-KOSAC-small

For this model, we used the Korean Comments Dataset¹ and the unlabeled part of the Korean Hate Speech dataset (Moon et al. 2020). Using the letters from those data names, we named this model with the prefix ‘CH’ (C for Comments and H for Hate Speech) and the suffix ‘small’, since we trained the model with a smaller scale, as with KR-BERT-KOSAC-small.

The data’s total size is about 12.7GB with 91M comments, and the sentiment features based on the single-morpheme tokens annotated in the KOSAC

¹https://github.com/Beomi/KcBERT/releases/tag/TrainData_v1

lexicon are infused into the input embeddings of this model. The vocabulary for this model is constructed based on the training data with the WordPiece model, consisting of 20,000 tokens.

We trained the CH-BERT-KOSAC-small model using an Adam optimizer, a max sequence length of 128, a training batch size of 64, and a learning rate of 1e-4, for 500K steps with two Tesla V100 GPUs for about 70 hours. Because the datasets used for this model do not have additional document segmentations and each comment is a document itself, we did not use the next sentence prediction task during training for this model.

Table 24: Training performances of CH-BERT-KOSAC-small

	CH-BERT-KOSAC-small	
global steps	500K	1M
MLM acc	52.40	54.02
MLM loss	2.4872837	2.356253

Table 24 reports the training performance of CH-BERT-KOSAC-small at the points of 500K and 1M steps. For this model, the table includes only MLM performance since we excluded the NSP task for model training.

With the parameter settings above, both sentiment-combined models are trained only for 500K steps. This is because, while training the models, their training losses drop remarkably around 500K steps. Figure 16 visualizes our CH-BERT-KOSAC-small model’s training losses up to 1M steps, which do not change much after the point of 500K steps. In the figure, the x-axis rep-

resents the number of steps, and the y-axis indicates the training loss. More steps of training are unlikely to efficiently improve the performance of the model for the training time.

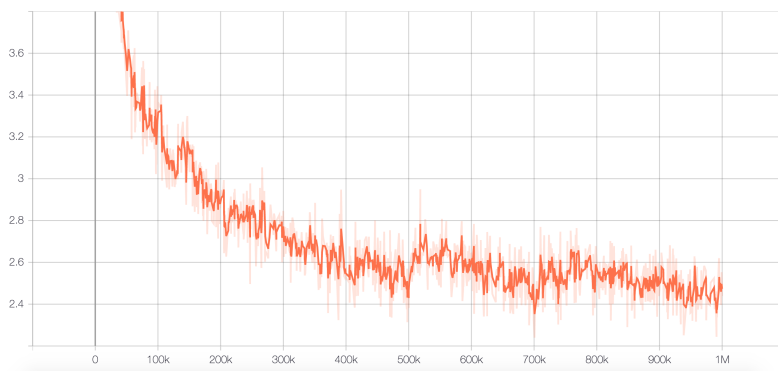


Figure 16: Training loss of CH-BERT-KOSAC-small

Both models have eight layers of encoders, eight attention heads for the self-attention mechanism, and a hidden size of 512, the same as those of medium-scale BERT-based models released by Google Research (Turc et al. 2019). However, we named the scale ‘small’ in this study to avoid confusion with the names of the other models presented, such as KR-BERT-MEDIUM.

Although a smaller-scaled model is proposed in Turc et al. (2019) originally for knowledge distillation in environments with limited computational resources, we applied this smaller scale of model to training models with sentiment features using less training time and hardware resources. The KR-BERT-KOSAC model described in 6.3 is a base-scale model with 12 encoder layers, 12 attention heads, and a hidden size of 768, which takes about 92

hours to train for 2M steps using a Google Cloud TPU. On the other hand, KR-BERT-KOSAC-small and CH-BERT-KOSAC-small are small-scale models using a maxlen of 128 that take about 70 hours to train using two GPUs. The fact that the model uses only GPUs is a clear advantage, because GPUs are more commonplace since they are much less expensive to use and easier to access than TPUs.

Since the existing generic pre-trained models are imported, there is no need for additional training for them, and the new smaller-scale models with sentiment features can be trained using less hardware in a shorter time for fewer steps. After that, only a fine-tuning process is required, achieved by applying both models to the downstream tasks simultaneously. The small-scale sentiment-combined models trained for 500K steps report an apparent increase in task performance when combined with existing general BERT-based models, as shown in 7.2.

7.2 Experiments and Results

We report the test results for the validation set and test set of two sentiment analysis tasks, the movie review corpus, NSMC, and the hate speech dataset, HSD, in Table 25. The ‘+’ operation between the two models means external fusing. For example, the row that includes ‘+ CH-BERT-KOSAC-small’ below the ‘KR-BERT-MEDIUM’ row reports the task performance of a combination of KR-BERT-MEDIUM and CH-BERT-KOSAC-small models.

As in Chapter 6, for the task datasets NSMC and HSD, we modified and

utilized the TensorFlow-based BERT fine-tuning codes released by Google Research. We used the Adam optimizer, a training batch size of 128, an evaluation batch size 8, a max sequence length of 128, and a learning rate of $5e-5$, and fine-tuned the models for five epochs over each dataset.

Table 25: Task performances of externally-fused models

	NSMC		HSD	
	dev acc	test acc	dev F1	test F1
M-BERT	87.08	86.82	73.39	52.03
+ KR-BERT-KOSAC-small	89.34	89.03	75.00	54.23
+ CH-BERT-KOSAC-small	89.39	89.18	80.22	55.63
KorBERT	90.48	89.81	78.74	54.33
+ KR-BERT-KOSAC-small	90.75	90.49	77.70	55.64
+ CH-BERT-KOSAC-small	91.00	90.68	79.34	57.49
KR-BERT	89.86	89.74	78.18	54.53
+ KR-BERT-KOSAC-small	90.51	90.12	78.97	55.52
+ CH-BERT-KOSAC-small	90.60	90.53	78.90	56.28
KR-BERT-MEDIUM	90.90	90.29	81.93	57.91
+ KR-BERT-KOSAC-small	91.21	90.89	82.25	57.74
+ CH-BERT-KOSAC-small	90.93	90.81	83.63	58.90

In general, and especially clearly in the models based on M-BERT, KorBERT, KR-BERT, and KR-BERT-MEDIUM, both sentiment analysis tasks show higher performance when external fusing methods are applied. In movie review classification (NSMC), when the sentiment-combined models are combined with M-BERT, they show about 3%p performance improvement. KorBERT and the two models we trained in this dissertation, KR-BERT and KR-BERT-MEDIUM, also work better with the sentiment-combined models. In

particular, when KR-BERT-MEDIUM is combined with KR-BERT-KOSAC-small, the accuracies produced are the highest ever released up to November 2020, with 91.21 for the validation set and 90.89 for the test set. We also obtained high accuracies above 90 in the cases where KR-BERT-MEDIUM is combined with CH-BERT-KOSAC-small, and KR-BERT is combined with either of the two sentiment-combined models.

For the hate speech detection task, the externally fused models show much larger improvements from the baseline models. For example, M-BERT, combined with the CH-BERT-KOSAC-small model, reports a validation set F1 score of 80.22, which is about 7%p higher than the baseline using M-BERT only (73.39). KorBERT performance also increased when combined with the sentiment-combined models. While the validation set F1 score slightly decreased when KR-BERT-KOSAC-small was attached KorBERT, the test set F1 score of the combination improved, and with the CH-BERT-KOSAC-small model added, both the validation and test sets reported increased F1 scores. KR-BERT and KR-BERT-MEDIUM also improved their F1 scores when combining the sentiment-combined models with them.

On the other hand, in HSD, there are slight differences in performance between using KR-BERT-KOSAC-small and CH-BERT-KOSAC-small models for external fusing with baseline models. When the CH-BERT-KOSAC-small model is combined, the performance improved abruptly compared to when the KR-BERT-KOSAC-small model is combined, or when the baseline model is used alone in general. In NSMC, the KR-BERT-KOSAC-small

and CH-BERT-KOSAC-small models do not show a significant difference in performance. This is most likely because the training data of the CH-BERT-KOSAC-small model consists of comments on news articles, and is therefore more similar to the Hate Speech dataset in stylistic and sentiment properties. Such comments are expected to include the stylistic properties of informal texts, and contain a writer’s subjectivity and emotion in the text owing to the domain characteristics. Therefore, a model trained with those texts would have an advantage in tasks of sentiment analysis.

The domain composition of KR-BERT-KOSAC-small’s training corpus is similar to other existing BERT-based models. Its training data include Wikipedia texts and news articles whose sentences are relatively formal and refined. On the other hand, most Korean sentiment tasks are classification problems on user-generated web comments, including more informal texts. Such text includes abbreviations, coinages, emoticons, spacing errors, and typos. Some of these characteristics function to express a writer’s subjectivity and emotion in the text. The comments data we used for CH-BERT-KOSAC-small consist of comments on news articles, which include these characteristics, and makes the data likely to include the commenter’s opinion and sentiment. The example sentences below represent such differences between formal and informal texts.

- (11) 22일 제주를 제외한 전국에서 미세먼지 비상 저감 조치가 발령된다.
22(isipi)il ceycwulul ceyoyhan cenkwukeyse misey menci pisang cekam
cochika pallyengtoynta.

“Emergency fine dust reduction measures will be issued across the country except for Jeju on the 22nd.”

(12) 볼매녀 념유쾌 잼나고 이빠요!!!♡

polmaynye nemyukhway caymnako ippeyo!!!♡

“The more I see her, the more attractive she is! She is so cheerful, funny, and pretty!!!♡”

Example (11) is an excerpt from a news article included in the training data of KR-BERT, while Example (12) is a sentence extracted from the hate speech dataset (Moon et al. 2020) used for sentiment analysis in this study. The former is written in very refined form to inform, and the latter is a comment freely written and expresses the judgment of a woman in a very informal language.

In Example (12), 볼매녀 polmaynye is an abbreviation of a phrase meaning “a woman who becomes more attractive as she continues to be seen,” and 념유쾌 nemyukhway is an abbreviated form of a phrase indicating “so delightful.” 잼나고 caymnako is an abbreviation meaning “[It’s] fun, and...”. These forms reveal the idea that the writer wants to represent in a compressed and effective manner. Special symbols such as ‘♡’ are also examples of a compressed means that indicate that the writer is friendly to the target.

These stylistic properties are hard to find in texts such as Wikipedia, books, and news articles, while they mainly appear in texts such as web corpora, messenger texts, and comment datasets written freely by the general public. So we constructed an additional model, CH-BERT-KOSAC-small, with train-

ing data only including informal texts. We suppose that this property of the corpora may help to perform sentiment-related tasks.

From the results above, we can conclude that the models fused with the sentiment-combined model perform better than the single baseline models. Additionally, training the sentiment-combined models on a smaller scale with fewer steps does not adversely affect how the performance increases and requires less training time and fewer resources.

7.3 Chapter Summary

In this chapter, we considered a novel method of infusing sentiment features to the sentiment analysis of BERT-based models while using less training time and hardware cost. The external fusing method we propose in this chapter combines an existing pre-trained BERT model with a sentiment-combined model we trained in the phase of fine-tuning to perform the tasks at a time. The former is a base-scaled general-purpose language model trained for many steps, and thus performs well in various NLP tasks. The latter is expected to function well in sentiment-related tasks using its sentiment features. Combining these two models with different advantages will allow them to interact and perform the task better.

In 7.2, we verified that the combination of models obtains better performance generally in sentiment-related tasks, such as movie review classification and hate speech detection. This external fusing method reduces overall processing time and hardware cost, since we import the existing generic pre-

trained models and train a small-scale sentiment-combined model for a few steps.

8 Conclusion

In this dissertation, we trained Korean-specific BERT-based sentence embedding models that include contextualized language representations. First, we composed a vocabulary based on 2.47GB of training data consisting of Korean Wikipedia texts and news articles, and trained a KR-BERT model to perform several Korean NLP tasks. We also trained one more Korean BERT-based model KR-BERT-MEDIUM based on training data expanded by adding comments data and legal texts to the existing KR-BERT training data.

We also composed sentiment features using polarity and intensity values assigned to each token in a sequence based on a Korean sentiment lexicon and reconstructed the BERT input embeddings, infusing the features into them. Accordingly, we trained a sentiment-combined model, KR-BERT-KOSAC, and applied it to sentiment analysis.

Additionally, we proposed a novel method to infuse sentiment features into BERT-based sentiment analysis. Through our external fusing method, a general-purpose pre-trained BERT model and a sentiment-combined model are imported respectively, and combined when fine-tuning in sentiment analysis tasks. For the generic pre-trained model, we used various BERT-based models such as M-BERT, KorBERT, and the models we trained in this dissertation. The sentiment-combined model includes the small-scale models, KR-BERT-KOSAC-small and CH-BERT-KOSAC-small, which we trained for fewer steps in Chapter 7. This method effectively and efficiently utilizes sentiment features requiring less training time and computation costs.

8.1 Summary of Contribution and Results

8.1.1 Construction of Korean Pre-trained BERT Models

We trained BERT-based models with training data composed of Korean texts that can be applied to several Korean NLP tasks, including Named Entity Recognition, Semantic Textual Entailment, Question Answering, and Sentiment Analysis. We obtained improved or comparable performances with our models in many of the tasks than those of other existing models.

Some of the implemented models (KR-BERT and KR-BERT-KOSAC) are released on GitHub so that users can easily access and utilize them when processing Korean texts. There are not many task datasets publicly available in Korean, and fewer of them are sufficient, though some of them are now beginning to be released. Thus, pre-trained models applied to a wide range of NLP tasks will help for low-resource tasks. Such models could also contribute to theoretical linguistics and Korean linguistics since they are gradually applying various NLP methodologies, including word embedding models.

8.1.2 Construction of a Sentiment-Combined Model

We built sentiment features with sentiment information annotated in the Korean Sentiment Analysis Corpus, KOSAC, and combined them with BERT input embeddings to compose a sentiment-combined model. Model comparisons demonstrated that they perform better in language modeling and sentiment analysis when sentiment features are infused.

We also observed the effects of both the polarity and intensity features through ablation studies on language modeling and sentiment analysis. By comparing the performances of ablated models, which do not contain either one or neither of the sentiment features, with those of KR-BERT-KOSAC, which contain both of the sentiment features, we confirmed that utilizing both polarity and intensity features is better than in other cases.

Moreover, the sentiment-combined model we implemented, KR-BERT-KOSAC, is released on GitHub for public access.

8.1.3 External Fusing of Two Pre-Trained Models to Gain Performance and Cost Advantages

In this dissertation, we proposed a method of externally fusing an existing pre-trained model and a sentiment-combined model. At this time, the two models are combined by loading different vocabularies and pre-trained checkpoint files, and summing the losses obtained by each model during the fine-tuning process.

In particular, new models with sentiment features can be implemented using less hardware and shorter training time by training with a smaller scale and fewer steps. Therefore, given a pre-trained model, the external fusing method only requires lightweight training of a sentiment-combined model, and a simple model combination in the fine-tuning process. It allows our novel sentiment fusing method to save on training time and effort. Additionally, in most pre-trained models, the externally combined models recorded improved per-

formance in sentiment analysis than when the pre-trained models performed alone.

8.2 Future Directions and Open Problems

8.2.1 More Training of KR-BERT-MEDIUM for Convergence of Performance

The KR-BERT-MEDIUM model using the expanded training data based on KR-BERT is trained for 2M steps, as with KR-BERT. However, the size of the expanded training dataset is more than ten times larger than that of KR-BERT. Therefore, more training steps for KR-BERT-MEDIUM would be necessary to observe whether the model reports improved language modeling and sentiment analysis performances. For example, HanBERT reported that the model is trained for more than 5M steps, and KoBERT has been trained for around 20M steps, according to their loss graphs.

While not included in this study, we compared the training performances and task results obtained at 1M and 2M steps of training KR-BERT-MEDIUM. The model showed a converged training loss value before 2M steps, between 1M and 2M steps, not showing a significant difference in performance between 1M and 2M steps. However, the model's visualized training losses are not very flat, including a little upward and downward movement, as in Figure 8.

The task performances also reported that the model worked better at 2M steps than at 1M steps in all the NLP tasks described in this dissertation. Therefore, it may be worth observing whether task performance improves and

converges while training the models for further steps. If sentiment-combined models are added to those improved models by external fusing, task performance would increase and be stable.

8.2.2 Observation of Changes Depending on the Domain of Training Data

We assumed first that as training data for a model grows in quantity and its composition of domains, the model’s task performance would increase. However, KR-BERT-MEDIUM had higher performance than KR-BERT only in a few tasks. Moreover, when KR-BERT-KOSAC-small and CH-BERT-KOSAC-small are combined with baseline pre-trained models in our external fusing methodology, performance increases are different from each other. Both have improved results, but in most cases, CH-BERT-KOSAC-small has a more significant performance increase. A common fact in these two observations is that the domains that make up the training data of the models being compared are different.

We supposed that a model trained with data that includes informal textual characteristics would work well in a task consisting of similar data. Much of the data added to KR-BERT-MEDIUM are based on informal texts, which is stylistically similar to user-generated texts, such as the NSMC and HSD datasets. Therefore, KR-BERT-MEDIUM performed better on NSMC and HSD than other tasks, and the CH-BERT-KOSAC-small model contributed more than the KR-BERT-KOSAC-small model in tasks.

It would be worth organizing models according to the training data domain

and observing how they work in different tasks. The result analysis will help determine the configuration of training data to implement an improved pre-training model.

8.2.3 Overlap of Sentiment Features with Linguistic Knowledge that BERT Learns

In this study, we infused additional sentiment features into BERT to better perform in sentiment-related tasks. However, contextualized embedding models such as ELMo and BERT learn various linguistic phenomena included in their training data themselves (Pennington et al. 2014). The linguistic phenomena learned by the models may already contain the sentiment information of each token and the contexts in which they were formed in a sequence, so that the sentiment features we constructed may have been redundant.

It is not yet clear whether this is the case, as it is a block box problem, since it is impossible to analyze such models by directly investigating the inside of the system. However, considering the performance improvement of the previous studies that infused the linguistic features into contextualized embedding models, we judged that utilizing additional features would help the model's function. Several studies have attempted to probe into how contextualized embedding models such as BERT internally capture linguistic phenomena in terms of its encoder layers and attention heads (Clark, Khandelwal, et al. 2019; Hewitt and Manning 2019; Jawahar et al. 2019). Referring to these studies, justifying the composition of features through such methodologies

would improve the models.

8.2.4 The Specific Process of Sentiment Features Helping the Language Modeling of BERT is Unknown

We supposed that the addition of sentiment features would help improve the model for the language modeling process during training and relevant task performances. However, this is an indirect judgment from the language modeling performances, and from the task results of models with additional sentiment features and the basic pre-trained models without them. The language modeling process is also an area of the black box problem that is impossible to judge and analyze directly. Nevertheless, this problem also should be resolved, considering different methods that can probe the impact of features during language modeling in as much detail as possible through indirect methods.

Bibliography

- Alsentzer, Emily et al. (2019). “Publicly Available Clinical BERT Embeddings”. In: *Proceedings of the 2nd Clinical Natural Language Processing Workshop*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, pp. 72–78.
- Antoun, Wissam, Fady Baly, and Hazem Hajj (2020). “AraBERT: Transformer-based Model for Arabic Language Understanding”. In: *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, p. 9.
- Baccianella, Stefano, Andrea Esuli, and Fabrizio Sebastiani (2010). “SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining”. In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*. European Language Resources Association (ELRA).
- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *Proc. of the 3rd ICLR*.
- Bao, Lingxian, Patrik Lambert, and Toni Badia (2019). “Attention and Lexicon Regularized LSTM for Aspect-based Sentiment Analysis”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Association for Computational Linguistics, pp. 253–259.
- Bar-Haim, Roy et al. (2006). “The second pascal recognising textual entailment challenge”. In: *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*. Vol. 6. 1. Venice, pp. 6–4.
- Beltagy, Iz, Kyle Lo, and Arman Cohan (2019). “SciBERT: Pretrained Language Model for Scientific Text”. In: *EMNLP*. eprint: arXiv:1903.10676.
- Benamara, Farah et al. (2007). “Sentiment analysis: Adjectives and adverbs are better than adjectives alone”. In: *Proceedings of ICWSM conference*.

- Bentivogli, Luisa et al. (2009). “The Fifth PASCAL Recognizing Textual Entailment Challenge.” In: *Proceedings of TAC 2009*.
- Bojanowski, Piotr et al. (2017). “Enriching Word Vectors with Subword Information”. In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146. ISSN: 2307-387X.
- Bowman, Samuel R. et al. (2015). “A large annotated corpus for learning natural language inference”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.
- Cambria, Erik et al. (2018). “SenticNet 5: Discovering Conceptual Primitives for Sentiment Analysis by Means of Context Embeddings”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, pp. 1795–1802.
- Cañete, José et al. (2020). “Spanish Pre-Trained BERT Model and Evaluation Data”. In: *PML4DC at ICLR 2020*.
- Cer, Daniel et al. (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)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1–14.
- Clark, Kevin, Urvashi Khandelwal, et al. (2019). “What Does BERT Look At? An Analysis of BERT’s Attention”. In: *BlackBoxNLP@ACL*.
- Clark, Kevin, Minh-Thang Luong, et al. (2020). “Pre-Training Transformers as Energy-Based Cloze Models”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.
- Collobert, Ronan et al. (Feb. 2011). “Natural Language Processing (Almost) from Scratch”. In: *Journal of Machine Learning Research* 12, pp. 2493–2537.

- Conneau, Alexis et al. (2018). “XNLI: Evaluating Cross-lingual Sentence Representations”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Dagan, Ido, Oren Glickman, and Bernardo Magnini (2005). “The PASCAL recognizing textual entailment challenge”. In: *Machine Learning Challenges Workshop*. Springer, pp. 177–190.
- Daniluk, Michał et al. (2017). “Frustratingly short attention spans in neural language modeling”. In: *arXiv preprint arXiv:1702.04521*.
- Devlin, Jacob et al. (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)*. Association for Computational Linguistics, pp. 4171–4186.
- Dolan, William B and Chris Brockett (2005). “Automatically constructing a corpus of sentential paraphrases”. In: *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.
- Fellbaum, Christiane (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.
- Fu, Peng et al. (2018). “Learning sentiment-specific word embedding via global sentiment representation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.
- Galassi, Andrea, Marco Lippi, and Paolo Torrioni (2020). “Attention in Natural Language Processing”. In: *IEEE Transactions on Neural Networks and Learning Systems*.
- Giampiccolo, Danilo et al. (2007). “The third pascal recognizing textual entailment challenge”. In: *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*. Association for Computational Linguistics, pp. 1–9.

- Gim, Eunyeong (2004). “A Study on the Korean Emotion Verb”. PhD thesis. Chonnam National University.
- Gururangan, Suchin et al. (2018). “Annotation Artifacts in Natural Language Inference Data”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, pp. 107–112.
- Ham, Jiyeon et al. (2020). “KorNLI and KorSTS: New Benchmark Datasets for Korean Natural Language Understanding”. In: *arXiv preprint arXiv:2004.03289*.
- Hewitt, John and Christopher D. Manning (2019). “A Structural Probe for Finding Syntax in Word Representations”. 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)*. Association for Computational Linguistics, pp. 4129–4138.
- Hinkle, Dennis E., William Wiersma, and Stephen G. Jurs (2003). *Applied Statistics for the Behavioral Sciences*. Houghton Mifflin.
- Howard, Jeremy and Sebastian Ruder (2018). “Universal Language Model Fine-tuning for Text Classification”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 328–339.
- Hu, Minqing and Bing Liu (2004). “Mining and summarizing customer reviews”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177.
- Huang, Kexin, Jaan Altosaar, and Rajesh Ranganath (2019). “ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission”. In: *arXiv:1904.05342*.
- Huang, Luyao, Chi Sun, et al. (2019). “GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge”. 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)*. Association for Computational Linguistics, pp. 3507–3512.
- Husein, Zolkepli (2018). *Malaya, Natural-Language-Toolkit library for bahasa Malaysia, powered by Deep Learning Tensorflow*. <https://github.com/huseinzol05/malaya>.
- Iyer, Shankar, Nikhil Dandekar, and Kornel Csernai (2017). *First Quora Dataset Release: Question Pairs*. URL: <https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs> (visited on 04/03/2019).
- Jawahar, Ganesh, Benoit Sagot, and Djamé Seddah (2019). “What Does BERT Learn about the Structure of Language?” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 3651–3657.
- Jiang, Nanjiang and Marie-Catherine de Marneffe (2019). “Evaluating BERT for natural language inference: A case study on the CommitmentBank”. 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)*. Association for Computational Linguistics, pp. 6086–6091.
- Ke, Pei et al. (2020). “SentiLARE: Sentiment-Aware Language Representation Learning with Linguistic Knowledge”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 6975–6988.
- Kikuta, Yohei (2019). *BERT Pretrained model Trained On Japanese Wikipedia Articles*. <https://github.com/yoheikikuta/bert-japanese>.
- Kłeczek, Dariusz (2020). “Polbert: Attacking Polish NLP Tasks with Transformers”. In: *Proceedings of the PolEval 2020 Workshop*. Institute of Computer Science, Polish Academy of Sciences.

- Kovaleva, Olga et al. (2019). “Revealing the Dark Secrets of BERT”. 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)*. Association for Computational Linguistics, pp. 4365–4374.
- Lan, Zhenzhong et al. (2020). “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations”. In: arXiv: 1909.11942 [cs.CL].
- Lee, Jinhyuk, Wonjin Yoon, et al. (2019). “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics*. ISSN: 1367-4803.
- Lee, Sangah, Hansol Jang, et al. (2020a). “A Small-Scale Korean-Specific BERT Language Model”. In: *Journal of KHISE* 47.7.
- (2020b). “KR-BERT: A Small-Scale Korean-Specific Language Model”. In: *ArXiv abs/2008.03979*.
- Lee, Sangah and Hyopil Shin (2020). “A Method of Infusing Additional Features into Pre-Trained BERT Models for Sentiment Analysis”. In: *Proceedings of Korea Software Congress 2020*.
- Levine, Yoav et al. (2020). “SenseBERT: Driving Some Sense into BERT”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 4656–4667.
- Li, Yingjie and Cornelia Caragea (2019). “Multi-Task Stance Detection with Sentiment and Stance Lexicons”. 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)*. Association for Computational Linguistics, pp. 6299–6305.
- Liu, Yinhan et al. (2019). “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *arXiv preprint arXiv:1907.11692*.

- Mikolov, Tomas et al. (2013). “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781*.
- Moon, Jihyung, Won Ik Cho, and Junbum Lee (2020). “BEEP! Korean Corpus of On-line News Comments for Toxic Speech Detection”. In: *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*. Association for Computational Linguistics, pp. 25–31.
- Nguyen, Dat Quoc and Anh Tuan Nguyen (2020). “PhoBERT: Pre-trained language models for Vietnamese”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1037–1042.
- Park, Cheoneum et al. (2019). “Korean Movie Review Sentiment Analysis using Self-Attention and Contextualized Embedding”. In: *Journal of KIISE* 46, pp. 901–908.
- Park, Jangwon (2020a). *KoBERT-KorQuAD*. <https://github.com/monologg/KoBERT-KorQuAD>.
- (2020b). *KoBERT-NER*. <https://github.com/monologg/KoBERT-NER>.
- (2020c). *KoELECTRA*. <https://github.com/monologg/KoELECTRA/tree/master/finetune>.
- Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 1532–1543.
- Peters, Matthew et al. (2018). “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pp. 2227–2237.
- Polignano, Marco et al. (2019). “AlBERTo: Italian BERT Language Understanding Model for NLP Challenging Tasks Based on Tweets”. In: *Proceedings of the*

Sixth Italian Conference on Computational Linguistics (CLiC-it 2019). Vol. 2481. CEUR.

- Qian, Qiao et al. (2017). “Linguistically Regularized LSTM for Sentiment Classification”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 1679–1689.
- Radford, Alec et al. (2018). “Improving language understanding with unsupervised learning”. In: *Technical report, OpenAI*.
- Rahate, Rohini S and M Emmanuel (2013). “Feature selection for sentiment analysis by using svm”. In: *International Journal of Computer Applications* 84.5, pp. 24–32.
- Rajpurkar, Pranav et al. (2016). “Squad: 100,000+ questions for machine comprehension of text”. In: *arXiv preprint arXiv:1606.05250*.
- Salton, Giancarlo, Robert Ross, and John Kelleher (2017). “Attentive language models”. In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 441–450.
- Sang, Tjong Kim, Erik F., and Fien De Meulder (2003). “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147.
- Sanh, Victor et al. (2019). “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv preprint arXiv:1910.01108*.
- Schuster, M. and K. Nakajima (2012). “Japanese and Korean voice search”. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152.

- Shin, Hyopil et al. (2012). “Annotation scheme for constructing sentiment corpus in Korean”. In: *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pp. 181–190.
- Socher, Richard et al. (2013). “Recursive deep models for semantic compositionality over a sentiment treebank”. In: *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642.
- Sukhbaatar, Sainbayar, Jason Weston, Rob Fergus, et al. (2015). “End-to-end memory networks”. In: *Advances in neural information processing systems* 28, pp. 2440–2448.
- Sun, Qingying et al. (2018). “Stance Detection with Hierarchical Attention Network”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, pp. 2399–2409.
- Tang, Duyu et al. (2014). “Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 1555–1565.
- Teng, Zhiyang, Duy-Tin Vo, and Yue Zhang (2016). “Context-Sensitive Lexicon Features for Neural Sentiment Analysis”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 1629–1638.
- Tian, Hao et al. (2020). “SKEP: Sentiment Knowledge Enhanced Pre-training for Sentiment Analysis”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 4067–4076.
- Turc, Iulia et al. (2019). “Well-Read Students Learn Better: On the Importance of Pre-training Compact Models”. In: arXiv: 1908.08962 [cs.CL].

- Turney, Peter D. (2002). “Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews”. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 417–424.
- Vaswani, Ashish et al. (2017). “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30, pp. 5998–6008.
- Virtanen, A. et al. (2019). “Multilingual is not enough: BERT for Finnish”. In: *ArXiv abs/1912.07076*.
- Vries, Wietse de, Andreas van Cranenburgh, and Malvina Nissim (2020). “What’s so special about BERT’s layers? A closer look at the NLP pipeline in monolingual and multilingual models”. In: *Findings of EMNLP*, pp. 4339–4350.
- Wang, Alex, Yada Pruksachatkun, et al. (2019). “SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems”. In: *arXiv preprint 1905.00537*.
- Wang, Alex, Amanpreet Singh, et al. (2018). “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, pp. 353–355.
- Warstadt, Alex, Amanpreet Singh, and Samuel R. Bowman (2019). “Neural Network Acceptability Judgments”. In: *Transactions of the Association for Computational Linguistics 7*, pp. 625–641.
- Wiebe, J., T. Wilson, and Claire Cardie (2005). “Annotating Expressions of Opinions and Emotions in Language”. In: *Language Resources and Evaluation 39*, pp. 165–210.
- Williams, Adina, 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)*. Association for Computational Linguistics, pp. 1112–1122.
- Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann (2005). “Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis”. In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 347–354.
- Wolf, Thomas et al. (2020). “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, pp. 38–45.
- Wu, Yonghui et al. (2016). “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144*.
- Yang, Zhilin et al. (2019). “XLNet: Generalized Autoregressive Pretraining for Language Understanding”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 5753–5763.
- Yoo, Won Joon (2020). *Introduction to Deep Learning for Natural Language Processing, Wikidocs*. URL: <https://wikidocs.net/book/2155> (visited on 12/28/2020).
- Zhang, Zhengyan et al. (2019). “ERNIE: Enhanced Language Representation with Informative Entities”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 1441–1451.
- Zhou, Junru et al. (2020). “LIMIT-BERT : Linguistics Informed Multi-Task BERT”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, pp. 4450–4461.

- Zhu, Yukun et al. (2015). “Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books”. In: *The IEEE International Conference on Computer Vision (ICCV)*.
- Zwillinger, Daniel and Stephen Kokoska (1999). *CRC standard probability and statistics tables and formulae*. Crc Press.

Appendices

A. Python Sources

A.1 Construction of Polarity and Intensity Embeddings

The polarity and intensity embeddings are constructed by the code below and then added to the existing input embeddings ‘output.’ The variable ‘output’ is the sum of input token embeddings, segment (token type) embeddings, and position embeddings. ‘polarity_ids’ and ‘intensity_ids’ are the vectors consisting of polarity and intensity indices, respectively, assigned to each token. The way of constructing the polarity embeddings and intensity embeddings is the same as that of BERT’s original segment embeddings.

Algorithm 1: Construction of polarity and intensity embeddings

```
import tensorflow as tf

...

input_shape = get_shape_list(input_tensor, expected_rank=3)
batch_size = input_shape[0]
seq_length = input_shape[1]
width = input_shape[2]

...

if use_polarity:
    if polarity_ids is None:
        raise ValueError(
            "polarity_ids must be specified if "
            "use_polarity is True.")
    polarity_table = tf.get_variable(
        name=polarity_embedding_name,
```

```

        shape=[polarity_vocab_size, width],
        initializer=create_initializer(initializer_range))

    flat_polarity_ids = tf.reshape(polarity_ids, [-1])
    polarity_one_hot_ids = tf.one_hot(flat_polarity_ids, depth=polarity_vocab_size)
    polarity_embeddings = tf.matmul(polarity_one_hot_ids, polarity_table)
    polarity_embeddings = tf.reshape(polarity_embeddings,
                                    [batch_size, seq_length, width])

    output += polarity_embeddings

    if use_intensity:
        if intensity_ids is None:
            raise ValueError("'intensity_ids' _must_be_specified_if"
                              "'use_intensity' _is_ True.")
        intensity_table = tf.get_variable(
            name=intensity_embedding_name,
            shape=[intensity_vocab_size, width],
            initializer=create_initializer(initializer_range))

        flat_intensity_ids = tf.reshape(intensity_ids, [-1])
        intensity_one_hot_ids = tf.one_hot(flat_intensity_ids, depth=intensity_vocab_size)
        intensity_embeddings = tf.matmul(intensity_one_hot_ids, intensity_table)
        intensity_embeddings = tf.reshape(intensity_embeddings,
                                         [batch_size, seq_length, width])

        output += intensity_embeddings

    ...

```

A.2 External Fusing of Different Pre-Trained Models

At the fine-tuning and testing phase for each NLP task, two pre-trained models and the tokenizers based on their vocabularies are loaded to process the task sequences. Each task sequence is tokenized and processed by two models separately to obtain pairs of token vectors (`input_ids` and `input_ids_kosac`). For

all the variables, the suffix ‘token’ is for the general-purpose BERT model, and the suffix ‘kosac’ is for the sentiment-combined model.

Algorithm 2: Construction of polarity and intensity embeddings

```
import tensorflow as tf

...

def create_model(bert_config_token, bert_config_kosac, is_training, input_ids, input_mask,
                 segment_ids, polarity_ids, intensity_ids, labels, num_labels, use_one_hot_embeddings,
                 input_ids_kosac, input_mask_kosac, segment_ids_kosac, polarity_ids_kosac,
                 intensity_ids_kosac):
    """Creates a classification model."""

    model_token = modeling_token.BertModel(
        config=bert_config_token,
        is_training=is_training,
        input_ids=input_ids,
        input_mask=input_mask,
        token_type_ids=segment_ids,
        use_one_hot_embeddings=use_one_hot_embeddings)

    model_kosac = modeling_kosac.BertModel(
        config=bert_config_kosac,
        is_training=is_training,
        input_ids=input_ids_kosac,
        input_mask=input_mask_kosac,
        token_type_ids=segment_ids_kosac,
        use_one_hot_embeddings=use_one_hot_embeddings,
        polarity_ids=polarity_ids_kosac,
        intensity_ids=intensity_ids_kosac)

    output_layer_token = model_token.get_pooled_output()
    output_layer_kosac = model_kosac.get_pooled_output()

    hidden_size_token = output_layer_token.shape[-1].value
```

```

hidden_size_kosac = output_layer_kosac.shape[-1].value

output_weights_token = tf.get_variable(
    "output_weights_token", [num_labels, hidden_size_token],
    initializer=tf.truncated_normal_initializer(stddev=0.02))

output_bias_token = tf.get_variable(
    "output_bias_token", [num_labels], initializer=tf.zeros_initializer())

output_weights_kosac = tf.get_variable(
    "output_weights_kosac", [num_labels, hidden_size_kosac],
    initializer=tf.truncated_normal_initializer(stddev=0.02))

output_bias_kosac = tf.get_variable(
    "output_bias_kosac", [num_labels], initializer=tf.zeros_initializer())

one_hot_labels = tf.one_hot(labels, depth=num_labels, dtype=tf.float32)

with tf.variable_scope("loss"):
    if is_training:
        # I.e., 0.1 dropout
        output_layer_token = tf.nn.dropout(output_layer_token, keep_prob=0.9)

    logits_token = tf.matmul(output_layer_token, output_weights_token, transpose_b=True
        )
    logits_token = tf.nn.bias_add(logits_token, output_bias_token)
    probabilities_token = tf.nn.softmax(logits_token, axis=-1)
    log_probs_token = tf.nn.log_softmax(logits_token, axis=-1)

    per_example_loss_token = -tf.reduce_sum(one_hot_labels * log_probs_token, axis
        =-1)
    loss_token = tf.reduce_mean(per_example_loss_token)

    if is_training:
        # I.e., 0.1 dropout
        output_layer_kosac = tf.nn.dropout(output_layer_kosac, keep_prob=0.9)

    logits_kosac = tf.matmul(output_layer_kosac, output_weights_kosac, transpose_b=

```

```
    True)
    logits_kosac = tf.nn.bias_add(logits_kosac, output_bias_kosac)
    probabilities_kosac = tf.nn.softmax(logits_kosac, axis=-1)
    log_probs_kosac = tf.nn.log_softmax(logits_kosac, axis=-1)

    per_example_loss_kosac = -tf.reduce_sum(one_hot_labels * log_probs_kosac, axis
        =-1)
    loss_kosac = tf.reduce_mean(per_example_loss_kosac)

    logits = logits_token + logits_kosac
    probabilities = probabilities_token + probabilities_kosac
    per_example_loss = per_example_loss_token + per_example_loss_kosac
    loss = loss_token + loss_kosac

    return (loss, per_example_loss, logits, probabilities)

...

```

B. Examples of Experiment Outputs

This section presents examples from the NLP task datasets we used in this study. For all the examples below, the model prediction labels are produced by KR-BERT fine-tuning.

Named Entity Recognition

Text:

기자: 체코 프라하에서 서향으로 4km 떨어진 작은 강호 와르자시내가 도처에서 모여든 집사람들로 떠들썩합니다 .

kica: cheykho phulahaeyse sehyangulo 4km ttelecin cakun kangho walucasinayka tocheeyse moyetun cipsalamtullo ttetulssekhapnita .

“Journalist: A small river lake, Warsasina, four kilometers west of Prague, the Czech Republic, is buzzing with housewives from all over the place.”

Gold Label: CVL-B LOC-B LOC-B O NUM-B O O O LOC-B O O O O

Model Prediction: CVL-B LOC-B LOC-B O NUM-B O O O O O O O O

Question Answering

Question:

엑스박스 360의 물품 중 별도로 판매되고 있는 것은?

eyksupaksu 360uy mwulphwum cwung pyeltolo phanmaytoyko issnun kesun?

“Which of the Xbox 360 items are sold separately?”

Gold Label: 액세서리 aykseyseli “accessories”

Model Prediction: 다양한 액세서리를 tayanghan aykseyselilul “a variety of accessories”

Natural Language Inference

Sentence 1:

그는 그들이 북쪽으로 올라갔다고 말했다.

kunun kutuli pwukccokulo ollakasstako malhayssta.

“He said they went up north.”

Sentence 2:

그는 그들이 남쪽으로 내려갔다고 했어요.

kunun kutuli namccokulo naylyekasstako haysseyo.

“He said they went south.”

Gold Label: Contradiction

Model Prediction: Contradiction

Semantic Textual Similarity

Sentence 1:

한 여자가 양파를 자르고 있다.

han yecaka yangphalul caluko issta.

“A woman is cutting onions.”

Sentence 2:

한 여자가 두부를 자르고 있다.

han yecaka twupwulul caluko issta.

“A woman is cutting tofu.”

Gold Label: 1.800

Model Prediction: 2.4379761

Movie Sentiment Classification

Text:

심심할때 보면 딱 좋은 영화
simsimhalttay pomyen ttak cohun yenghwa
“a perfect movie to watch when bored”

Gold Label: Positive

Model Prediction: Positive

Hate Speech Detection

In the case of Hate Speech Detection, the gold labels for the test dataset are not released. The test performance for this task is only obtained by the official leaderboard.

Text:

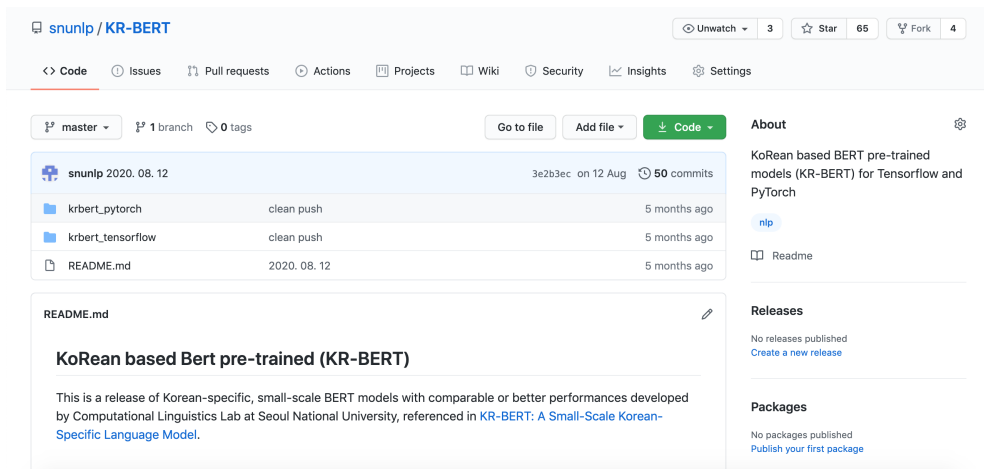
선남선녀 너무 이쁘다^^
sennamsennyne nemwu ipputa^^
“Good looking men and women. They are so pretty. ^^”

Model Prediction: None

C. Model Releases through GitHub

KR-BERT

<https://github.com/snunlp/KR-BERT>



snunlp / KR-BERT

Unwatch 3 Star 65 Fork 4

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

File	Commit	Time
krbert_pytorch	clean push	5 months ago
krbert_tensorflow	clean push	5 months ago
README.md	2020. 08. 12	5 months ago

README.md

KoRean based Bert pre-trained (KR-BERT)

This is a release of Korean-specific, small-scale BERT models with comparable or better performances developed by Computational Linguistics Lab at Seoul National University, referenced in [KR-BERT: A Small-Scale Korean-Specific Language Model](#).

About
KoRean based BERT pre-trained models (KR-BERT) for Tensorflow and PyTorch
[nlp](#)
Readme

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

KR-BERT-KOSAC

<https://github.com/snunlp/KR-BERT-KOSAC>

snunlp / KR-BERT-KOSAC

Unwatch 2 Star 5 Fork 3

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

Go to file Add file Code

snunlp Update README.md 49c5307 4 minutes ago 9 commits

img	2020. 06. 11	7 months ago
tensorflow	2020. 06. 18	6 months ago
README.md	Update README.md	4 minutes ago

README.md

KR-BERT-KOSAC

A pretrained Korean-specific BERT model including sentiment features to perform better at sentiment-related tasks, developed by Computational Linguistics Lab at Seoul National University.

It is based on our character-level [KR-BERT](#) models which utilize WordPiece and BidirectionalWordPiece tokenizers.

About

Expanded KR-BERT for Sentiment Analysis

nlp sentimental-analysis

Readme

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

국문초록

한국어 사전학습모델 구축과 확장 연구: 감정분석을 중심으로

최근 트랜스포머 양방향 인코더 표현 (Bidirectional Encoder Representations from Transformers, BERT) 모델에 대한 관심이 높아지면서 자연어처리 분야에서 이에 기반한 연구 역시 활발히 이루어지고 있다. 이러한 문장 단위의 임베딩을 위한 모델들은 보통 학습 과정에서 문장 내 어휘, 통사, 의미 정보를 포착하여 모델링한다고 알려져 있다. 따라서 ELMo, GPT, BERT 등은 그 자체가 다양한 자연어처리 문제를 해결할 수 있는 보편적인 모델로서 기능한다.

본 연구는 한국어 자료로 학습한 단일 언어 BERT 모델을 제안한다. 가장 먼저 공개된 한국어를 다룰 수 있는 BERT 모델은 Google Research의 multilingual BERT (M-BERT)였다. 이는 한국어와 영어를 포함하여 104개 언어로 구성된 학습 데이터와 어휘 목록을 가지고 학습한 모델이며, 모델 하나로 포함된 모든 언어의 텍스트를 처리할 수 있다. 그러나 이는 그 다중언어성이 갖는 장점에도 불구하고, 각 언어의 특성을 충분히 반영하지 못하여 단일 언어 모델보다 각 언어의 텍스트 처리 성능이 낮다는 단점을 보인다. 본 연구는 그러한 단점들을 완화하면서 텍스트에 포함되어 있는 언어 정보를 보다 잘 포착할 수 있도록 구성된 데이터와 어휘 목록을 이용하여 모델을 구축하고자 하였다.

따라서 본 연구에서는 한국어 Wikipedia 텍스트와 뉴스 기사로 구성된 데이터를 이용하여 KR-BERT 모델을 구현하고, 이를 GitHub을 통해 공개하여 한국어 정보처리를 위해 사용될 수 있도록 하였다. 또한 해당 학습 데이터에 댓글 데이터와 범조문과 판결문을 덧붙여 확장한 텍스트에 기반해서 다시 KR-BERT-MEDIUM

모델을 학습하였다. 이 모델은 해당 학습 데이터로부터 WordPiece 알고리즘을 이용해 구성된 한글 중심의 토큰 목록을 사전으로 이용하였다. 이들 모델은 개체명 인식, 질의응답, 문장 유사도 판단, 감정 분석 등의 다양한 한국어 자연어처리 문제에 적용되어 우수한 성능을 보고했다.

또한 본 연구에서는 BERT 모델에 감정 자질을 추가하여 그것이 감정 분석에 특화된 모델로서 확장된 기능을 하도록 하였다. 감정 자질을 포함하여 별도의 임베딩 모델을 학습시켰는데, 이때 감정 자질은 문장 내의 각 토큰에 한국어 감정 분석 코퍼스(KOSAC)에 대응하는 감정 극성(polarity)과 강도(intensity) 값을 부여한 것이다. 각 토큰에 부여된 자질은 그 자체로 극성 임베딩과 강도 임베딩을 구성하고, BERT가 기본으로 하는 토큰 임베딩에 더해진다. 이렇게 만들어진 임베딩을 학습한 것이 감정 자질 모델(sentiment-combined model)이 된다.

KR-BERT와 같은 학습 데이터와 모델 구성을 유지하면서 감정 자질을 결합한 모델인 KR-BERT-KOSAC를 구현하고, 이를 GitHub을 통해 배포하였다. 또한 그로부터 학습 과정 내 언어 모델링과 감정 분석 과제에서의 성능을 얻은 뒤 KR-BERT와 비교하여 감정 자질 추가의 효과를 살펴보았다. 또한 감정 자질 중 극성과 강도 값을 각각 적용한 모델을 별도 구성하여 각 자질이 모델 성능 향상에 얼마나 기여하는지도 확인하였다. 이를 통해 두 가지 감정 자질을 모두 추가한 경우에, 그렇지 않은 다른 모델들에 비하여 언어 모델링이나 감정 분석 문제에서 성능이 어느 정도 향상되는 것을 관찰할 수 있었다. 이때 감정 분석 문제로는 영화평의 공부정 여부 분류와 댓글의 악플 여부 분류를 포함하였다.

그런데 위와 같은 임베딩 모델을 사전학습하는 것은 많은 시간과 하드웨어 등의 자원을 요구한다. 따라서 본 연구에서는 비교적 적은 시간과 자원을 사용하는 간단한 모델 결합 방법을 제시한다. 적은 수의 인코더 레이어, 어텐션 헤드, 적은

임베딩 차원 수로 구성된 감정 자질 모델을 적은 스텝 수까지만 학습하고, 이를 기존에 큰 규모로 사전학습되어 있는 임베딩 모델과 결합한다. 기존의 사전학습 모델에는 충분한 언어 모델링을 통해 다양한 언어 처리 문제를 처리할 수 있는 보편적인 기능이 기대되므로, 이러한 결합은 서로 다른 장점을 갖는 두 모델이 상호작용하여 더 우수한 자연어처리 능력을 갖도록 할 것이다. 본 연구에서는 감정 분석 문제들에 대한 실험을 통해 두 가지 모델의 결합이 학습 시간에 있어 효율적이면서도, 감정 자질을 더하지 않은 모델보다 더 정확한 예측을 할 수 있다는 것을 확인하였다.

키워드: 한국어 문장 임베딩 모델, BERT, 언어 모델링, 멀티 헤드 셀프 어텐션, 감정 자질, 모델 외적 결합, 감정 분석

학번: 2016-30038