문학석사 학위논문

Linguistically Explicit BERT with Part-of-Speech Information

품사 임베딩 정보를 결합한 언어학적 BERT 모델

2021 년 2 월

서울대학교 대학원

언어학과 언어학전공

백 연 미

Linguistically Explicit BERT with Part-of-Speech Information

품사 임베딩 정보를 결합한 언어학적 BERT 모델

지도 교수 신 효 필

이 논문을 문학석사 학위논문으로 제출함

2020 년 11 월

서울대학교 대학원

언어학과 언어학전공

백 연 미

백연미의 문학석사 학위논문을 인준함

2021 년 1 월

위 원 장 남 승 호 (인)

부위원장 신 효 필 (인)

위 원 유 현 조 (인)

Abstract

Linguistically Explicit BERT with Part-of-Speech Information

Baik, Yunmee

Department of Linguistics

The Graduate School

Seoul National University

This study incorporates part-of-speech, one of the most well-known linguistic features, to the input embedding of the BERT model to enhance the ability of the language model and investigates what linguistic knowledge the model learns from pre-training. Although BERT shows powerful performance on many downstream tasks of Natural Language Processing, many studies have reported that injecting explicit linguistic knowledge improves the performance of the BERT model. Also, several studies have inspected the linguistic representation encoded in BERT using probing classifiers. Probing task on the Korean dataset, however, has not yet been conducted.

In this study, we fuse POS embedding to the input embedding of the BERT model by (1) adding POS embedding to the BERT embedding(addPOS), (2) multiplying and then adding it to the input embedding(multiaddPOS), and (3) masking the POS of the masked token

while adding it to the input representation(maskPOS) in pre-training. We use Korean Wikipedia and news data as a corpus and MeCab POS tagger as a POS tagger and a tokenizer. In fine-tuning, we conduct 5 Korean downstream tasks (NSMC, NER, KorQuaD, KorNLI, KorSTS). As a result, the proposed POS models, especially the maskPOS model, show better performance on the tasks than the base MeCab-tokenized model which does not fuse POS information. In comparison to the state-of-the-art models, however, the POS models show low performance on the tasks.

We conduct a linguistic analysis of the maskPOS model. To identify syntactic information encoded in the model, the structural probe (Hewitt and Manning, 2019) is adapted on Korean datasets. The probe results show that the proposed POS model embeds syntax trees, encoding linguistic knowledge in its word representations. Further experiments are conducted for better performance of the POS models on the downstream task. We conclude that there is a possibility for improving the POS models.

This study suggests new methods to fuse linguistic information to the Korean pre-trained BERT model, and to the best of our knowledge, it is the first study to use “probe” on Korean datasets with the Korean-specific model. In this study deep learning architectures and linguistic theory are integrated, suggesting directions for future Korean NLP research.

Keywords: Natural Language Processing, Language Modeling, BERT, Word Embeddings, Part-of-Speech, Interpretability, Probe, Parse Tree

Student Number: 2018-20037

Table of Contents

List of Figures

List of Tables

1. Introduction

The Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2018) has demonstrated powerful performances on a wide range of tasks in natural language processing (NLP). This improvement of the model on the downstream tasks is because BERT can learn lexical, syntactic, and semantic information of a sentence (Clark et al., 2019; Coenen et al., 2019; Goldberg, 2019; Hewitt and Manning, 2019; Tenney et al.,2019).

Although BERT itself can capture linguistic information, the incorporation of linguistic information does help the language model for the related downstream task (Lee et al., 2020; Liu et al., 2019; Strubell et al., 2018; Sundararaman et al., 2019; Wang et al., 2019, Zhang et al., 2020; Zhou et al., 2020). In this study, we incorporate part-of-speech (POS) tags using MeCab POS tagger[1] in the KoNLPy[2] package into the Korean-specific BERT model from pre-training.

Part-of-speech is also known as word classes or syntactic categories which represent morphosyntactic information. It is one of the most familiar and simplistic linguistic feature. It gives information about the relationship between words by their distribution, such as adjectives describe nouns. Also, it informs a syntactic structure of a sentence (verbs are part of verb phrases), and a word sense disambiguation (Korean 가 *ka* as a particle or a verb). It is

[1] http://eunjeon.blogspot.com/

[2] https://konlpy.org/ko/latest/
KoNLPy is a Python package for natural language processing of the Korean language.

a useful feature in parsing, named entity recognition, information extraction, or coreference resolution (Jurafsky, 2019).

This study leverages part-of-speech (POS) tags using MeCab POS tagger to the BERT model from pre-training and runs experiments on the Korean downstream tasks. We implement the MeCab POS tagger which is known to be the most time-efficient and does not decompose Hangul (character) into Jamo (sub-character). We suggest new training methods by infusing this linguistic knowledge to the embeddings fed into the pre-trained model. The suggested models prove the effectiveness of adding linguistic features compared to the base MeCab model.

We implement linguistic probing tasks on the proposed POS model. “Probes” interpret how well linguistic knowledge is encoded in the model using various linguistic tasks. This study conducts the syntactic probing task of Hewitt and Manning (2019) to analyze the effect of adding POS information on the language model.

The outline of this research is as follows: Chapter 2 provides a literature review of embeddings, existing Transformer models with linguistic knowledge, and model interpretation. Chapter 3 provides a detailed account of Transformer architectures and the BERT model. Chapter 4 introduces POS models, which are BERT models with POS-infused embeddings. Chapter 5 describes the pre-training data, training details, and results. It also contains the following fine-tuning experiments and their results. Five Korean downstream tasks will be introduced in chapter 5. Chapter 6 analyzes the proposed models using the probing tasks to evaluate the models’ linguistic ability to learn the underlying structure of a sentence and conducts additional experiments to enhance the model’s performance on the downstream task. Chapter 7 concludes the study by providing a summary of the work and

discussing a limitation of our research and possible avenues for future research.

2. Literature Review

This chapter discusses the history and development of embeddings, and relevant literature review concerning Transformer models (Vaswani et al., 2017) with linguistic information, and model interpretation using probing tasks.

2.1. Embeddings

Word embeddings are vectors that represent words or the process that words are embedded in a particular vector space. It is based on the idea of vector semantics: representations of the meaning of words can be learned directly from their distributions. It is also based on the distribution hypothesis, defining a word by the distribution it occurs in the text, as the philosopher Ludwig Wittgenstein said “the meaning of a word is its use in the language”. If the two words occur in similar environments or distributions, they are likely to have a similar meaning.

Embeddings have been developed from embedding words to embedding sentences. The most famous word embedding algorithms are Word2Vec (Mikolov et al., 2013a), GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2017).

The Word2Vec model has two model architectures, CBOW and Skip-gram (Mikolov et al., 2013b). The Continuous Bag of Words (CBOW) model predicts the target word using context words. The Skip-gram model predicts the context words given a current target word. In general, the Skip-gram model is known to show better performance results for the downstream tasks

than the CBOW model.

Word2Vec implements negative sampling (Mikolov et al., 2013a) to reduce computation costs. A positive sample refers to a pair of a target word and surrounding context words, and a negative sample refers to a pair of a target word and a word that randomly samples in the vocabulary.

The GloVe model is trained on aggregated global word-word co-occurrence statistics from a corpus. A global word-word co-occurrence matrix organizes how frequently word pairs occur in a given corpus and matrix factorization is applied to approximate the matrix. The FastText model considers a subword, treating each word as a Bag of Character n-grams so that rare words get a good representation.

Like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), sentence embedding models offer an advantage over word embedding models, because while a word has a fixed embedding under word embedding models disregarding the context it appears, sentence embedding models produce context embeddings that dynamically differ depending on the meaning of the word in a sentence.

Embeddings are key aspects in transfer learning as embeddings with accurate representations of words or sentences in pre-training result in high performances on the downstream tasks in fine-tuning. Models applying transfer learning, the Transformer architecture and the BERT model, will be described in detail in Chapter 3.

2.2. Models with Linguistic Information

Recently, linguistic knowledge has been incorporated into the pre-

trained language model. Linguistically Informed Self-Attention (LISA) (Strubell et al., 2018) suggests the potential of incorporating linguistic information into the Transformer model by explicit modeling of syntax. Strubell et al. (2018) enhanced the performance of the model with multi-task learning across various syntactic knowledge, such as dependency parsing, POS tagging, predicate detection, and Semantic Role Labeling (SRL).

Zhou et al, (2020) also proposed the multi-task learning model LIMIT-BERT, proving the effectiveness of injecting linguistic knowledge into the BERT model. In addition to leveraging multi-task learning across POS tags, constituent and dependency syntactic parsing, span and dependency SRL, they introduced Syntactic and Semantic Phrase Masking. Syntactic Phrase Masking masks tokens in a constituent of a sentence together ([MASK] [MASK] [MASK] sells paper and wood products.), and Semantic Phrase Masking masks tokens in a dependency relation such as the predicate and its object argument (federal paper board [MASK] paper and wood [MASK].) This study also uses a masking strategy for the BERT model to learn linguistic information.

Other studies explicitly incorporated the model with one of the linguistic features. Syntax-Infused Transformer and BERT models (Sundararaman et al., 2019) infused POS information into the models using WordPiece tokenizer (Wu et al., 2016) and SpaCy[3] for POS tagger. For constituent information, Tree Transformer (Wang et al., 2019) proposed "Constituent Attention" to make the attention heads learn tree structures. Zhang et al. (2020) suggested SemBERT by incorporating pre-trained SRL information, and Liu et al. (2019) implemented a knowledge graph to get

[3] https://spacy.io/

sememe information.

This paper leverages POS information, a simple and familiar linguistic feature. Since POS forms a distribution, we believe that a model can learn a representation according to the distribution hypothesis. Furthermore, it will give useful information on an agglutinative language like Korean.

One of the proposed models has the same structure proposed in Sundararaman et al. (2019). The input token $h'_m = e_m + f_m^P$ where e_m is the BERT token embedding and f_m^P is POS embedding for token m. This study, however, suggests new methods other than just adding embeddings. Instead of SpaCy in Sundararaman et al. (2019), MeCab is used in this paper expecting to be more suitable for morphologically rich languages. The suggested models will be introduced in chapter 4.

2.3. Interpretation of Linguistic Knowledge of a Model

"Probes" are used to interpret how and what type of linguistic information is encoded in the model. This process is referred to as a "probing task" (Conneau et al., 2018), "diagnostic classifier" (Giulianelli et al., 2018), or an "auxiliary prediction task" (Adi et al., 2016). Such probes are conducted on various linguistic tasks and on different levels of the model.

There are studies concerning attention heads of the Transformer architecture. In Clark et al. (2019), specific attention heads in BERT showed particular linguistic phenomena. Kovaleva et al. (2019) suggested 5 patterns of attention heads using a heatmap, and that the "heterogeneous" pattern can be interpreted linguistically. Visualization tools are also proposed in Vig (2019), and Hoover et al. (2019).

On the layers of the model, Tenney et al. (2019) proved that syntactic knowledge appeared in initial layers and semantics in later layers using probing tasks such as POS, constituents, dependencies, entities, SRL, and coreference resolutions.

Hewitt and Manning (2019) proposed syntactic analysis using probing. They evaluated syntactic knowledge of the models by recovering syntactic dependencies in Penn Treebank (Marcus et al., 1993) from the models' token embeddings. Through a linear transformation of a model's word representation space, it was proved that the dependency trees were embedded in the model. In addition to the results, Coenen et al. (2019) visualized syntactic and semantic information of the models. We replicate the structural probing task of Hewitt and Manning (2019) using the Korean dataset. This will be demonstrated in chapter 6.

3. Transformer Architectures

This chapter will provide a description of Transformer architecture (Vaswani et al., 2017) and the BERT model (Devlin et al., 2018) as we use it as the base model architecture of the proposed POS models which will be described in the following chapter.

3.1. Transformer

Transformer architecture (Vaswani et al., 2017) is composed of a stack of encoders and decoders and when it takes an input sentence, it gets an output sentence in another (Figure 1). The BERT model (Devlin et al., 2018), which will be described in the next section, only uses the encoder part of the Transformer.

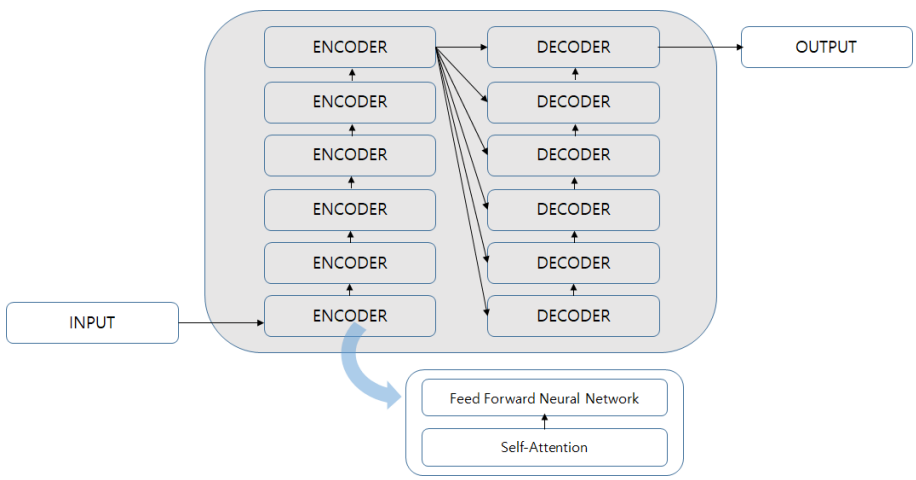


Figure 1. Transformer architecture, based on The Illustrated Transformer[4]

[4] http://jalammar.github.io/illustrated-transformer/

Each encoder has two sub-layers as in Figure 1: a self-attention layer and a feed-forward neural network. The outputs of the self-attention layer are fed to the feed-forward network. A list of vectors is taken as input and these vectors are passed into a self-attention layer, then through a feed-forward neural network, to the next encoder.

Multi-Head Attention (Equation 2) is a concatenation of multi Scaled Dot-Product Attention which is also called as Self-Attention (Equation 1). All equations in this section are from Vaswani et al. (2017).

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

The shape of an input embedding *X* of Self-Attention is the number of words in an input sentence and the dimension of the input embedding. A Query matrix *Q*, a Key matrix *K*, and a Value matrix *V* are created by multiplying the input matrix *X* and three weight matrices W^{Q}, W^{K}, and W^{V}. Then we take the dot product of the query vector with the key vector and divide it by the square root of d_{k} (the dimension of the key vectors) for scaling the variance, then normalize it through a softmax. The resulting score is multiplied by each value vector as the weighted value vectors and summed up.

$$\begin{aligned} \text{MultiHead}(\text{Q}, \text{K}, \text{V}) &= Concat(head_1, \dots, head_h)W^O \\ \text{where } \text{head}_\text{i} &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2}$$

In Multi-Head Attention, the Self-Attention is calculated *h* different times with *h* different weight matrices. The resulting *h* attention heads are

concatenated and multiplied with a weight matrix W^O so that the shape of the output is the same as that of the input. W_i^Q, W_i^K, W_i^V, and W^O are parameter matrices. By doing Self-Attention, the Transformer model can give attention to all the word pairs in a sentence considering the context without gradient vanishing problem.

The results send to the feed-forward neural network, using ReLU as an activation function (Equation 3). Instead of ReLU, GeLU is adopted in the BERT model.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (3)$$

To keep the order of the words in the input sequence, the Transformer uses positional encoding. BERT model, however, does not implement the same positional encoding. It uses the position embeddings instead. The input representation of the BERT model will be described in detail below.

3.2. Bidirectional Encoder Representations from Transformer (BERT)

Bidirectional Encoder Representations from Transformer (BERT) (Devlin et al., 2018) is a method that pre-trains language representations. As it adopts transfer learning, there are two procedures (Figure 2), pre-training and fine-tuning.

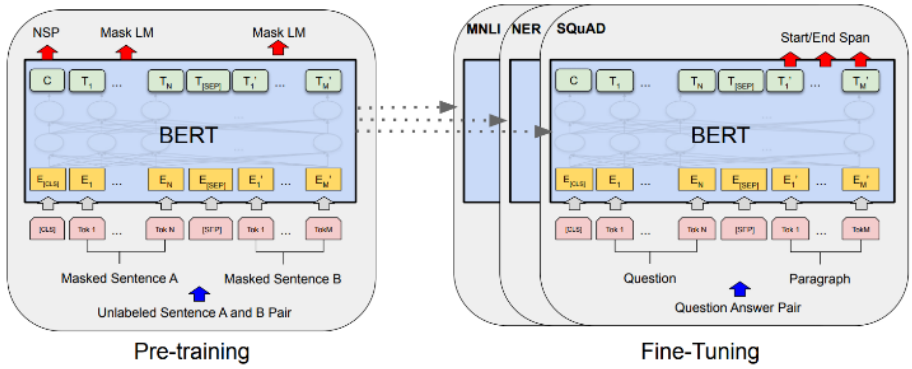


Figure 2. Pre-training and fine-tuning procedures for BERT from Devlin et al. (2018)

At pre-training, a general-purpose language understanding model is trained on a large text corpus like Wikipedia. It is an unlabeled and unsupervised process. The pre-trained model is fine-tuned and utilized for downstream NLP tasks like question answering, named entity recognition (NER), and natural language inference (NLI). A detailed description of each downstream task will be in chapter 5.

BERT is a Transformer Encoder stack. It has two model sizes, BERT-base and BERT-large. The BERT-base model has 12 encoder layers, 768 hidden embedding sizes, and 12 attention heads. The BERT-large model has 24 encoder layers, 1024 hidden embedding sizes, and 16 attention heads. In this study, we adopted BERT-base model architecture.

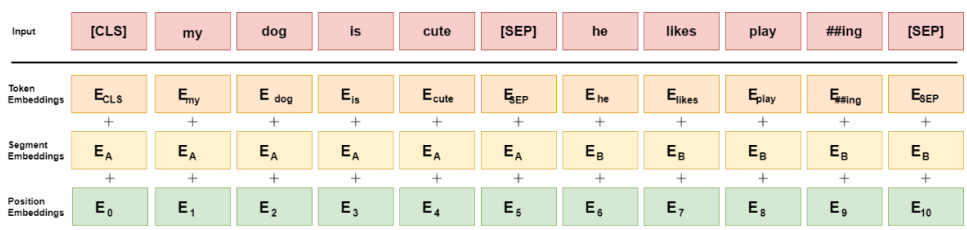


Figure 3. BERT input representation from Devlin et al. (2018)

BERT input representation, as in Figure 3, is the sum of the token embeddings, the segment embeddings, and the position embeddings which substitute the positional encoding of Transformer. WordPiece tokenizer (Wu et al., 2016) is used to token embeddings. Sentence pairs are packed into a single input sequence as the first token of the sequence is a classification token ([CLS]) and the two sentences are separated by a separation token ([SEP]). Segment embeddings indicate whether a token is from sentence A or sentence B. Position embeddings inform the position of the token in an input sequence.

POS models, proposed in this study, have one more input embedding, POS embeddings, to add to the input representation. The token of BERT is tokenized by WordPiece tokenizer, however, MeCab is used as the tokenizer for the convenience of adding POS embeddings in this study. The description of POS models will be provided in detail in the next chapter.

To train a deep bidirectional representation and understand sentence relationships, BERT has two training approaches: masked language model (MLM) and next sentence prediction (NSP).

In the MLM task, the model masks some percentage of the input tokens, usually 15%, at random, and predicts the masked tokens. Specifically, for the masked token t, 80% of the time the token t is replaced with [MASK] token, 10% of the time with a random token, and 10% of the time it remains unchanged.

In the NSP task, two sentences A and B are given for each training example, and the model predicts whether B is the actual next sentence that follows A (labeled as IsNext). 50% of the time, it is a random sentence from the corpus (labeled as NotNext). We utilize both tasks in the experiments.

4. Part-of-Speech Models

In this chapter, POS models with modified embedding methods are fed into the BERT architecture (Devlin et al., 2018) will be introduced. 3 embedding methods are suggested in this chapter: addPOS, multiaddPOS, and maskPOS model. MeCab POS tagger, instead of WordPiece (Wu et al., 2016), is implemented as a tokenizer, therefore the base model in the experiment (chapter 5) is the MeCab-tokenized BERT model (Figure 4).

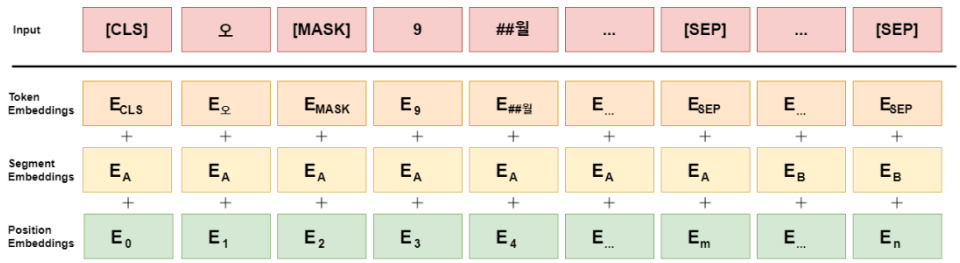


Figure 4. MeCab-tokenized model (base) input representation

By adding POS embeddings, we expect the model to learn the distribution of POS information, and thus understand the underlying syntactic representation in a language. We prove the effectiveness of incorporating linguistic knowledge into the input embeddings through the comparison of the POS models and the base model without POS embeddings in chapter 5.

4.1. Model Structure (Input Representation)

The input representation of POS models is a modification of BERT (Devlin et al., 2018) input representation, adding POS embeddings to the input embeddings of BERT. We propose 3 embedding methods here.

4.1.1. addPOS

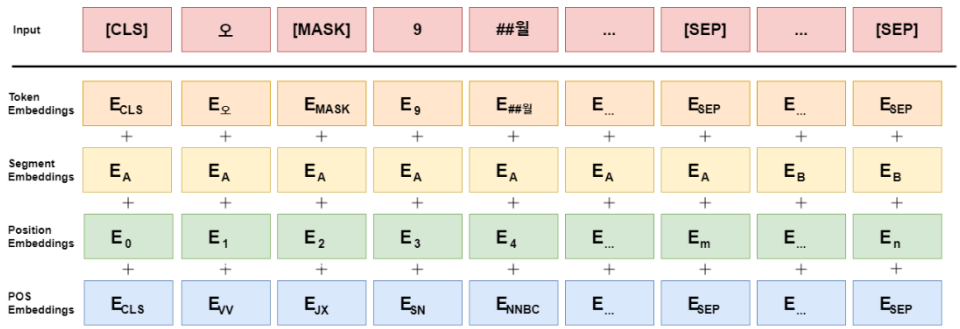


Figure 5. addPOS model input representation

The input representation of the addPOS model is composed of the BERT input embeddings, token embeddings, segment embeddings, and position embeddings, and POS embeddings tagged from the MeCab POS tagger. It is the same architecture with Syntax-infused BERT (Sundararaman et al., 2019), but the input sentence in this model is tokenized by the MeCab POS tagger instead of the WordPiece algorithm.

For the token m, the input representation h_m becomes Equation 4, where e_m is the token embedding of m, s_m is the segment embedding of m, p_m is the position embedding of m, and f_m is the POS embedding of m.

$$h_m = e_m + s_m + p_m + f_m \tag{4}$$

4.1.2. multiaddPOS

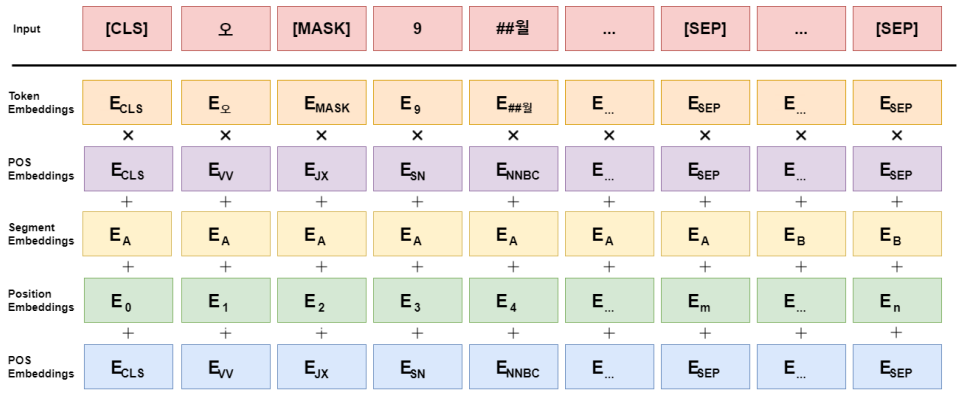


Figure 6. multiaddPOS model input representation

The multiaddPOS model input embeddings are the sum of the token embeddings multiplied by the POS embeddings, the segment embeddings, the position embeddings, and the POS embeddings. The two POS embeddings can have different initialization. For the token m, we compute the input representation h_m as:

$$h_m = e_m * f'_m + s_m + p_m + f_m \quad (5)$$

where e_m is the token embedding of m, s_m is the segment embedding of m, p_m is the position embedding of m, and f'_m and f_m is the POS embedding of m with different initialization.

4.1.3. maskPOS

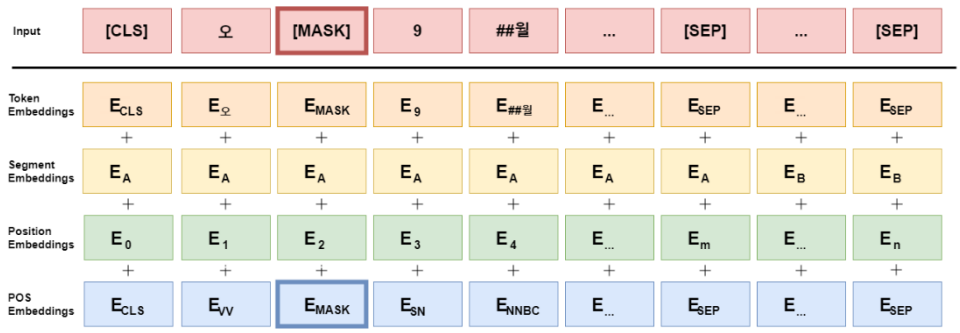


Figure 7. maskPOS model input representation

The maskPOS model has the same input representation as the addPOS model, adding the token embeddings, the segment embeddings, the position embeddings, and the POS embeddings, but the MLM task is applied to both the token embeddings and the POS embeddings. In the MLM task in the maskPOS model, if the *i*-th token is chosen, the token embedding and the POS embedding corresponding to the masked token t_i assign [MASK] embedding values respectively. The original token and POS will be predicted using cross-entropy loss, thus having two losses and accuracies for each embedding. The pre-training results will be presented in the next chapter.

5. Experiments

With the POS models described in chapter 4, pre-training and fine-tuning experiments were conducted to demonstrate the effectiveness of adding POS information, and the data for pre-training, training details, and results are presented below. The maskPOS model shows higher performances compared to the base model on 5 Korean downstream tasks, showing the language model with linguistic knowledge learns the underlying grammatical representation of a language. To the best of our knowledge, this is the first study to corporate POS knowledge to a pre-trained Korean BERT model.

5.1. Pre-training

Data, tokenizer, vocabulary, POS tag vocabulary, and other training details during pre-training will be described in this section.

5.1.1. Data

For the pre-training corpus, preprocessed 2.47GB Korean Wikipedia was used. It consists of 20M sentences, or 233M words. It is the same corpus on which KR-BERT (Lee et al., 2020) was pre-trained on. KR-BERT, however, implemented a WordPiece tokenizer (Wu et al., 2016).

5.1.2. Tokenizer

Unlike the original BERT model which utilized the WordPiece tokenizer, we adapted the MeCab POS tagger as a tokenizer of the models. The input sentences in the training examples are tokenized by the MeCab tokenizer. It makes the summation of the token embeddings and the POS

embeddings simpler because the token units and the POS units from the tagger correspond.

5.1.3. Vocabulary

After tokenizing the corpus with the MeCab tokenizer, we extracted 30,000 vocabularies in order of frequency. Then 5 special tokens that the BERT model uses were added: [CLS] which informs the start of a sentence, [SEP] which informs the end of a sentence, [PAD] for padding, [UNK] for unknown tokens, and [MASK] for masking in MLM tasks. Thus, the vocabulary of size 30,005 was used for pre-training the models.

5.1.4. Part-of-Speech Tag Vocabulary

POS tag vocabulary was proposed in two ways: *fulltag* and *endtag*. Because the POS tagging by MeCab is presented as a combination of agglutinative POS tags, it could be offered in many ways. *Fulltag* uses all the POS tags presented. *Endtag* uses only the last POS of tagging in order to capture particles in Korean and make the size of the vocabulary smaller. So for the example in Figure 8, MeCab POS tagging for token 그럴만도 *gurelmando* ‘It could be’ is VA+ETM+JX+JX. In the fulltag vocabulary, it uses all the combination VA+ETM+JX+JX, while in the endtag vocabulary, it only uses the last tag, JX. Other kinds of combinations have been experimented and the results are presented in chapter 6.

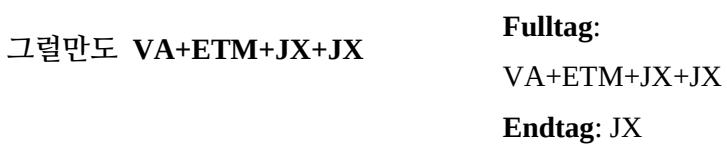


Figure 8. Two ways of POS tag vocabulary combination

5.1.5. Training Details

The pre-training batch size was 16, and the learning rate was 1e-4, then decreased to 2e-5 when the pre-training step was over 1M. 4 TITAN RTX (24GB RAM) GPUs were used for pre-training. Other hyperparameters are the same as the original BERT[5].

For the maskPOS with fulltag model, which shows the highest performances on the downstream tasks, we increased its batch size to 256, and pre-trained the model to 2M steps on TPU. It can be expected as the batch size increases, the performance of the model improves since the model can take more sentences as input at once. The learning rate was 1e-4.

5.2. Pre-training Results

Pre-training results of the base model and the proposed POS models are demonstrated below (Table 1). Because the NSP accuracy always shows almost 1, we exhibit only the MLM accuracy. All the models are showing high MLM accuracies.

[5] https://github.com/google-research/bert

Model		100k	500k	1M	2M step
MeCab	base	0.5945	0.6737	0.7097	0.7297
addPOS	fulltag	0.702	0.7886	0.8152	0.8197
	endtag	0.7006	0.7498	0.7452	0.783
multiaddPOS	fulltag	0.5465	0.7643	0.7734	0.7944
	endtag	0.506	0.7348	0.7654	0.7595
maskPOS	fulltag	(POS) 0.8032 (token) 0.5911	(POS) 0.8478 (token) 0.6769	(POS) 0.8642 (token) 0.7046	(POS) 0.8703 (token) 0.7182
	endtag	(POS) 0.8142 (token) 0.6047	(POS) 0.8523 (token) 0.6727	(POS) 0.8660 (token) 0.6988	(POS) 0.8816 (token) 0.7256

Table 1. Pre-training results of the base model and POS models. MLM accuracies are represented.

For the maskPOS with fulltag model, we increase its batch size from 16 to 256 as described in 5.1.5. The MLM accuracy of the two models is presented below. We can discover the MLM accuracy enhances as the batch size of the same model increases.

Model		100K	1M	2M
maskPOS (fulltag)	Batch size=16	(POS) 0.8032 (token) 0.5911	(POS) 0.8642 (token) 0.7046	(POS) 0.8703 (token) 0.7182
	Batch size=256 @TPU	(POS) 0.8242 (token) 0.6583	(POS) 0.8648 (token) 0.7240	**(POS) 0.8755 (token) 0.7400**

Table 2. Pre-training results of the maskPOS (fulltag) models with different batch sizes. MLM accuracies are represented.

5.3. Downstream Tasks

We conducted 5 Korean downstream tasks: NSMC (NAVER Sentiment Movie Corpus)[6], NER (Named Entity Recognition)[7], KorQuAD (Korean Question Answering Dataset)[8], KorNLI (Korean Natural Language Inference), and KorSTS (Korean Semantic Textual Similarity)[9] (Ham et al., 2020). The tasks and the metrics used for evaluating them will be described below.

[6] https://github.com/e9t/nsmc

[7] http://air.changwon.ac.kr/?page_id=10

[8] https://korquad.github.io/

[9] https://github.com/kakaobrain/KorNLUDatasets

5.3.1. Tasks

NSMC NAVER Sentiment Movie Corpus (NSMC) is a movie review dataset with the binary label for the sentiment class of the review (0: negative, 1: positive). It has 200K reviews, with half of the review is positive and the other half negative.

NER Named Entity Recognition (NER) is a dataset to extract named entities such as a person, organization, or location name. It has 14 categories.

KorQuAD The Korean Question Answering Dataset (KorQuAD) 1.0 is a dataset for Machine Reading Comprehension. It is composed of 20K QA pairs, finding an answer from 1~2 paragraphs.

KorNLI Korean Natural Language Inference (KorNLI) is the dataset for Korean Natural Language Understanding (KorNLU). Given a pair of two sentences, the task of KorNLI is to determine whether they are entailment, contradiction, or neutral. It has 950K examples.

KorSTS Korean Semantic Textual Similarity (KorSTS) is the dataset for KorNLU. The task of KorSTS is to determine the similarity between those two sentences from 0 to 5, given a pair of two sentences. KorSTS has 8K examples.

5.3.2. Evaluation Metrics

Accuracy, F1 score, Exact Match (EM), and Spearman's rank correlation coefficient are used to evaluate the results.

		Actual Class	
		True	**False**
Predicted Class	**True**	True Positive	False Positive
	False	False Negative	True Negative

Table 3. Confusion matrix

Accuracy (Equation6) and F1 score (Equation7) can be derived from a confusion matrix (Table 3). Accuracy is the rate of correct predictions out of the total number of true labels.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6}$$

F1 score is the weighted mean of precision and recall. EM is the number of exactly correct answers with the same start and end index.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1\ score = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} \tag{7}$$

Spearman's rank correlation coefficient or Spearman correlation shows the relationship between the rank values of two variables. It assesses monotonic relationships while the Pearson correlation between two variables

assesses linear relationships. Packages from scikit-learn[10], SciPy[11], and seqeval[12] are implemented as evaluation metrics for evaluating the performance of downstream tasks.

5.4. Downstream Task Results

The table below demonstrates the downstream task results of the base model and the POS models. It is seen in Table 4, the maskPOS models clearly outperform the base MeCab model. Among the models with a batch size of 16, the maskPOS model with fulltag shows the highest outputs. So we trained the model with a batch size of 256 as described in 5.1.5 for the maskPOS with fulltag model. It is observed in Table 4, the maskPOS (*fulltag*) model with a batch size of 256 surpasses the other models, especially the base MeCab-tokenized model without the POS embeddings. Therefore, we can conclude that when linguistic information is added to the pre-trained model, it is learning the underlying grammatical representation via the enhanced performance on the Korean downstream tasks.

[10] https://scikit-learn.org/stable/
Scikit-learn is simple and efficient tools for predictive data analysis.

[11] https://www.scipy.org/
SciPy (pronounced “Sigh Pie”) is open-source software for mathematics, science, and engineering.

[12] https://github.com/chakki-works/seqeval
seqeval is a Python framework for sequence labeling evaluation.

Model	5 epochs	NSMC	NER	KorQuad (dev)	KorNLI	KorSTS
Batch size 16, 2M step		Accuracy	F1	EM/F1	Accuracy	Spearman
MeCab	base	84.81	75.08	49.70, 80.68	72.89	0.7434
addPOS	fulltag	84.9	75.04	48.35, 79.41	74.59	0.7402
	endtag	84.84	74.7	47.81, 77.71	73.67	0.7326
multiaddPOS	fulltag	84.62	74.05	48.96, 79.77	73.59	0.7267
	endtag	84.46	74.11	48.04, 79.03	74.41	0.737
maskPOS	fulltag	84.98	75.48	49.46, 80.84	75.5	0.7501
	endtag	84.92	75.51	48.97, 80.40	74.67	0.7488
	fulltag, Batch size=256	**85.66***	**76.29**	**51.92, 83.42**	**76.6**	**0.7643***

Table 4. Downstream task results of the base model and POS models. Results marked with an asterisk implemented prediction batch size of 128.

However, as seen in Table 5, the POS models exhibit poor performances in comparison to other models with WordPiece tokenizer, KR-BERT, or State-of-the-Art models. The reason for this poor improvement of the POS embeddings will be discussed in the next section.

Model		NSMC	NER	KorQuad (dev)	KorNLI	KorSTS
		Accuracy	F1	EM/F1	Accuracy	Spearman
maskPOS (fulltag)	Batch size =16	84.98	75.48	49.46, 80.84	75.5	0.7501
	Batch size =256	85.66*	76.29	51.92, 83.42	76.6	0.7643*
KR-BERT (char)		89.74	85.77	72.04, 89.45	77.76	0.7746
State-Of-The-Art (KoELECTRA-Base etc.)	5+ epochs	Different test datasets.		**84.34, 92.58** (v2)	**80.89** (HanBERT)	**0.843** (v2)

Table 5. Downstream task results of maskPOS (fulltag) models and other state-of-the-art models. Results marked with an asterisk implemented prediction batch size of 128. The results on the state-of-the-art models are from KoELECTRA GitHub[13]. They used different test datasets on NSMC and NER tasks.

5.5. Analysis

The results captured in the previous section will be analyzed in this section. In 5.5.1, for the better performance that the maskPOS models showed in comparison to the other POS models, we drew a correlation heatmap

[13] https://github.com/monologg/KoELECTRA

between POS embeddings in order to figure out how a model learns a latent representation during pre-training. But for the poor performance of the POS models, we will point out the limitations of the current POS models in 5.5.2 and suggest the need for a further study (chapter 6).

5.5.1. Correlation Heatmap

To evaluate how POS models learn a latent representation during training, the correlation between POS embeddings is drawn to a heatmap (Figure 9).

As the performances on the downstream tasks in section 5.4 reflect, only the maskPOS models show correlation. The others remain yellow on the heatmap, meaning they learn no correlations between the POS embeddings. The maskPOS models seem to capture an underlying grammatical relationship between POS tags. It is in line with SpanBERT (Joshi et al., 2020) in that the heatmap proves the helpfulness of the MLM task.

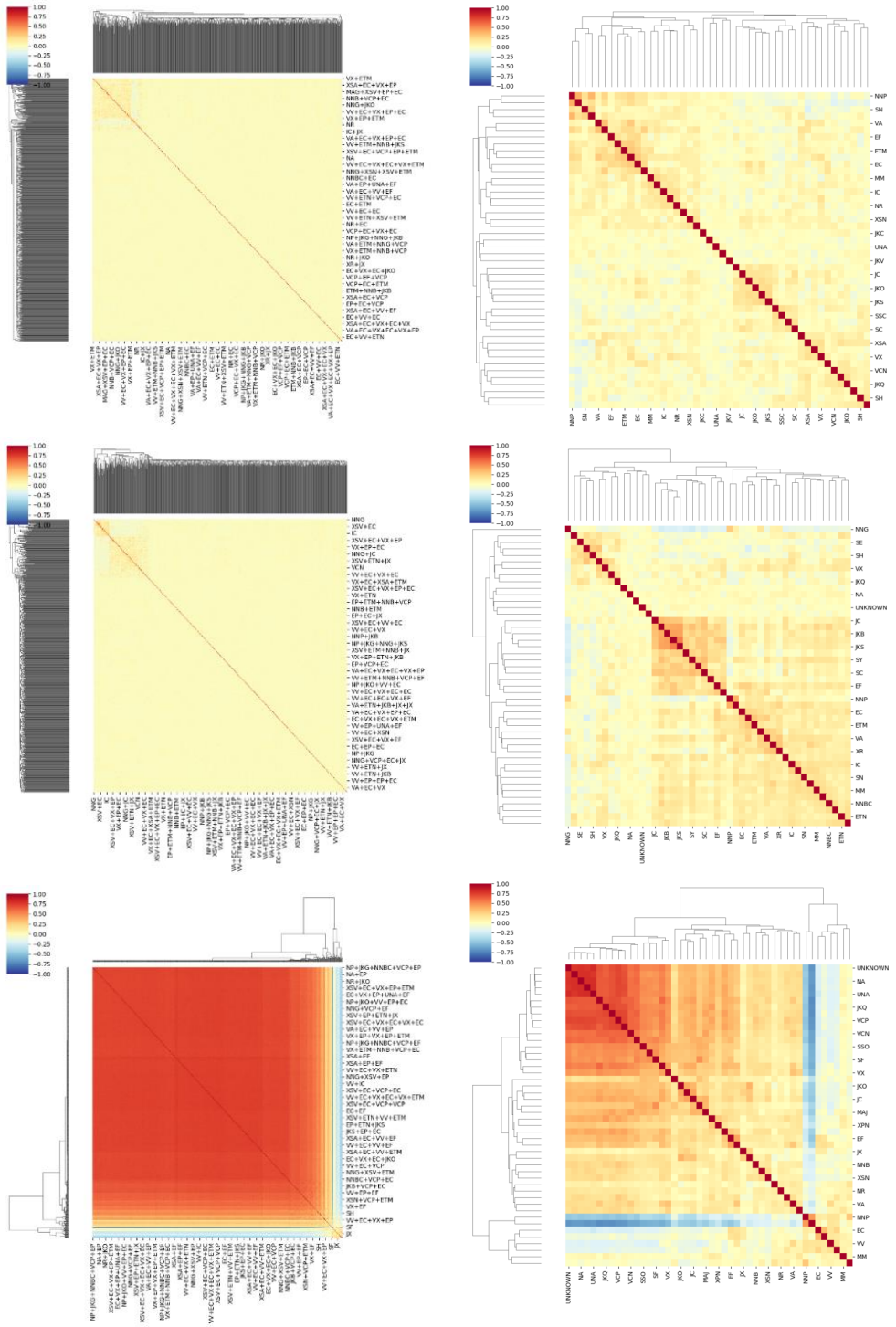


Figure 9. Heatmap representing the correlation between POS embeddings

5.5.2. Limitation

In order to explain the poor outcome of the POS models compared to the other models with WordPiece tokenizer (Wu et al., 2016) and the state-of-the-art models, the limitations of the current POS models will be identified here.

First, POS tagging is not off-the-shelf. POS tags are called from a pickle file, consist of a dictionary of a token and its POS tag. During the process of saving the dictionary file, only a high-frequency POS tag was selected, and others were neglected. For example, a token ##는 *##nun* 'is' can be tagged as JX, ETM, ETN, or VV, but since the most frequent POS tag is JX, the token always get the POS embedding of JX. In a phrase like 오 ##는 9 ##월 *o ##nun 9 ##wel* 'this September', a token *##nun*, which is ETM in the phrase, wrongly tagged as JX in Figure 5-7 because it is the most frequent POS of *##nun*. The possible solution for this problem will be discussed in 6.2.1 and 6.2.3.

##는 Counter({'**JX**': 7292984, 'ETM': 4944041, 'ETN': 5, 'VV': 1})

Second, the lack of vocabulary could reduce the models' performances. The pre-training corpus consists of 1M tokens, but we only use the vocabulary of the size of 30K concerning the computational capacity of the hardware. In 6.2.2, we will adjust the vocabulary size to deal with this limitation.

In addition, there could be some problem in preprocessing the corpus or the MeCab tokenizer to being an input of BERT architecture compared to the WordPiece tokenizer which is an unsupervised tokenization method based

on the frequency of a token in a corpus.

Further analysis to overcome the limitations above will be suggested in the next chapter.

6. Linguistic Analysis

Based on the experiment results in chapter 5, we conduct a linguistic analysis to evaluate the ability of the proposed POS models of learning syntactic representation by using a probing task (6.1). To further overcome the limitations in 5.5.2, further analysis on the POS tag combination, vocabulary size, and off-the-shelf POS tagging method will be used in the next section. We analyze the maskPOS model (fulltag, 30k vocabulary size, 16 batch size) since it showed comparably stable performances on the pre-training and fine-tuning downstream tasks in the previous chapter.

6.1. Syntactic Probing Analysis

Hewitt and Manning (2019) evaluated a neural network's representation by retrieving syntax trees embedded in a linear transformation of a model's word embedding space. We adapt the transformation to our POS model, maskPOS, to evaluate the linguistic ability of our model. This syntactic probing task will prove the effectiveness of adding morphosyntactic information to a model to make the model learn an underlying syntactic representation of a sentence.

6.1.1. The Structural Probe

The probe is based on the hypothesis that there is a linear transformation (an inner product) of the word representation that parse trees are embedded. Under the linear transformation B, vector distance under the inner product B^TB is supposed to encode parse trees. Given a parse tree, Hewitt and Manning (2019) consider a squared vector distance (an L2

distance) $\|h_i - h_j\|_B^2$ as a tree distance, and a norm $\|w_i\|$ as a parse depth where *i, j* are indexes of a word *w* in a sentence. The parse depth is the number of edges between the root of the parse tree and a word.

Equation 8 from Hewitt and Manning (2019) is a squared distance where $\mathrm{h}_i^{\mathrm{l}}$ is a vector representation of the *i*-th word of a sentence *l,* under the linear transformation of the word representation *Bh*. A parse depth can be defined as Equation 9.

$$d_B\left(h_i^l, h_j^l\right)^2 = \left(B\left(h_i^l - h_j^l\right)\right)^T \left(B\left(h_i^l - h_j^l\right)\right) \quad (8)$$

$$\|h_i\|_B^2 = (Bh_i)^T (Bh_i) \quad (9)$$

The matrix *B* is the parameters of the probe. It is trained to recreate the distance or the depth. In the case of the tree distance across all sentences T^l in a training corpus, it is approximated through Equation 10 (Hewitt and Manning, 2019) where $|s^l|$ is the length of the sentence *l*. Each sentence is normalized by the number of word pairs $|s^l|^2$.

$$\min_B \sum_l \frac{1}{|s^l|^2} \sum_{i,j} \left| d_{T^l}\left(w_i^l, w_j^l\right) - d_B\left(h_i^l, h_j^l\right)^2 \right| \quad (10)$$

6.1.2. Experiment Details

The probe requires CoNLL-formatted data. The KAIST Korean Universal Dependency Treebank[14] (Chun et al., 2018) was implemented to train the probe. The training dataset contains 23010 sentences, the dev dataset has 2066 sentences, and the test dataset has 2287 sentences (total 27363

[14] https://github.com/UniversalDependencies/UD_Korean-Kaist

sentences).

We conduct the probing analysis on the maskPOS model (fulltag, batch size of 16). Because the model was trained with TensorFlow (Abadi et al., 2016), we converted the TensorFlow checkpoint for maskPOS in a PyTorch save file. All configurations follow the original experiment[15].

6.1.3. Probe Evaluation Metrics

The predicted tree distance is evaluated on undirected attachment score (UUAS) and "distance Spearman (DSpr.)". UUAS is the percent of correct edges against the gold tree, evaluating tree reconstruction. To construct "DSpr.", the Spearman correlations between the gold tree and the predicted tree distance are averaged over all sentences of the same length and then averaged across sentence lengths 5-50.

To evaluate how well the predicted tree depth rebuilds the true tree, we report "root%" and "norm Spearman (NSpr.)". We assess the percentage of the correctly predicted root of the sentence which is the least deep word as "root%". "NSpr." metric replaces the "DSpr." metric with the Spearman correlation between the gold depth order of the word and the predicted ordering. All the evaluation metrics follow Hewitt and Manning (2019).

[15] https://github.com/john-hewitt/structural-probes

6.1.4. Probe Results

	Distance		Depth	
Layer	UUAS	DSpr.	Root%	NSpr.
1	63.29	0.6985	70.52	0.7223
2	63.28	0.6986	70.76	0.7248
3	63.29	0.6995	70.76	0.7247
4	63.28	0.6989	**71.39**	0.7221
5	**63.51**	0.6988	70.43	0.7246
6	63.31	0.6989	71.06	0.7225
7	63.44	0.6996	70.67	0.7240
8	**63.48**	**0.7001**	71.06	0.7246
9	63.21	0.6992	70.81	0.7231
10	63.29	0.6999	70.62	**0.7277**
11	63.34	0.6985	70.76	0.7260
12	63.41	**0.7002**	70.91	0.7249

Table 6. The probe results of parse tree distance and depth on maskPOS

The experiment results of parse tree distance probes and depth probes are reported in Table 6. There is no big difference between layers. It can be interpreted that the model encodes the parse tree stably across all the layers. Among the layers, the result of layer 8 will be reported since it shows comparably high performance on the probing task.

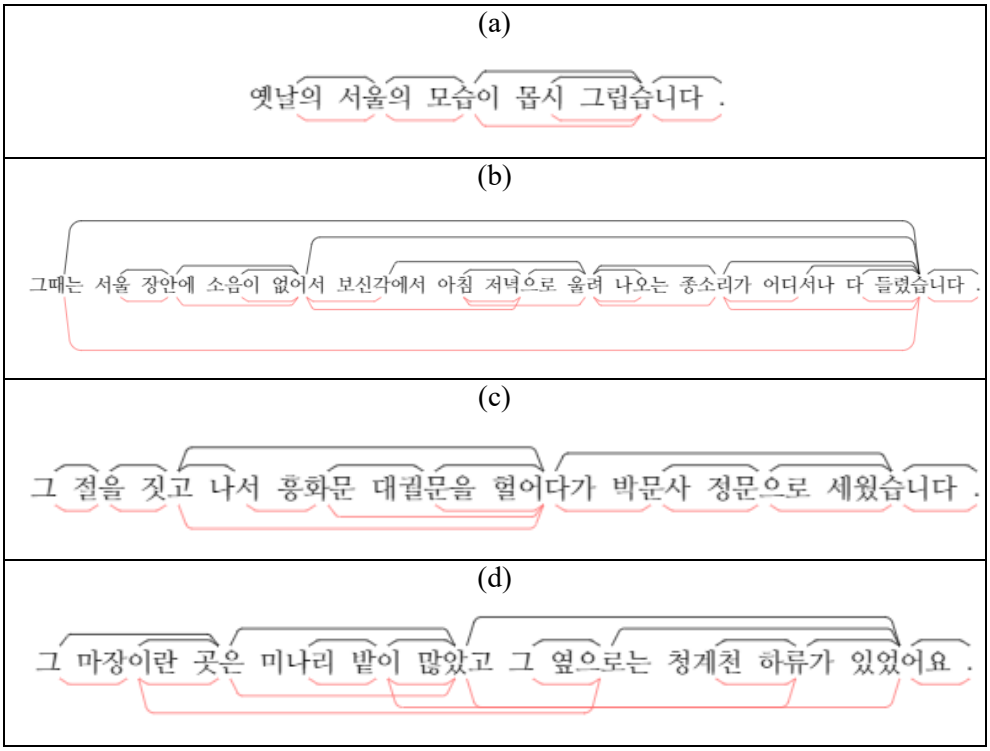


Figure 10. The gold parse trees (black) and the minimum spanning trees of predicted squared distances on maskPOS (red)

We present the gold parse trees (black) and the predicted parse trees on the maskPOS model (red) in Figure 10. Regarding a short and simple structured sentence such as Figure 10-a, the model perfectly predicts the parse tree. For Figure 10-b, the model predicts correctly the long-distance dependency between 그때는 *guttaynun* 'at that time' and 들렸습니다 *tullyesssupnita* 'heard'. Some adverb phrases are wrongly predicted, however, since Korean is a scrambling (Ross, 1967) language, adverb phrases can have flexible word order in Korean.

나서 *nase* in Figure 10-c is incorrectly presented on the predicted tree. MeCab POS tagger tags the token as VV but it seems to be used as an auxiliary. It seems POS tagging is not specific enough to reflect conjugations in Korean.

On the other hand, ##다가 *##taga* in 헐어다가 *heletaga* ‘demolish’ (Figure 10-c) and ##고 *##go* in 많았고 *manhassgo* ‘many’ (Figure 10-d) is EC in MeCab POS tagging. It is a naïve classification as determining what kind of syntactic phrase they are is a critical question in Syntax.

Figure 12 displays the depth in parse tree encoded in the gold tree (black) and the predicted tree on the maskPOS model (red) by vector norm after the linear transformation. The root of the parse tree, the least deep word, is generally identified correctly on the maskPOS model. Although the depth of the predicted tree does not perfectly correspond to that of the true tree, patterns, or relative depth, shown in the predicted trees are relatively similar to those of the true trees.

Figure 11 demonstrates all distances between all word pairs in a sentence. Short distances are visualized in darker colors and long distances are in lighter colors. According to Hewitt and Manning (2019), this is the rich structure in a parse distance matrix.

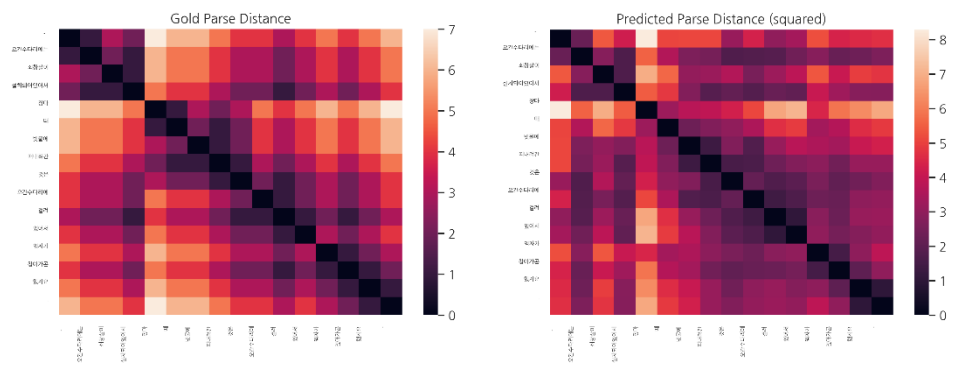
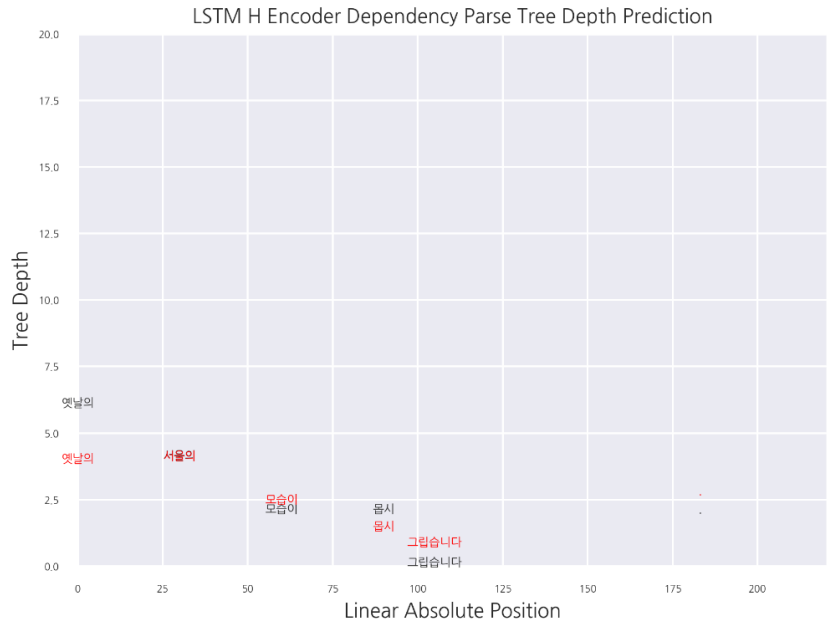
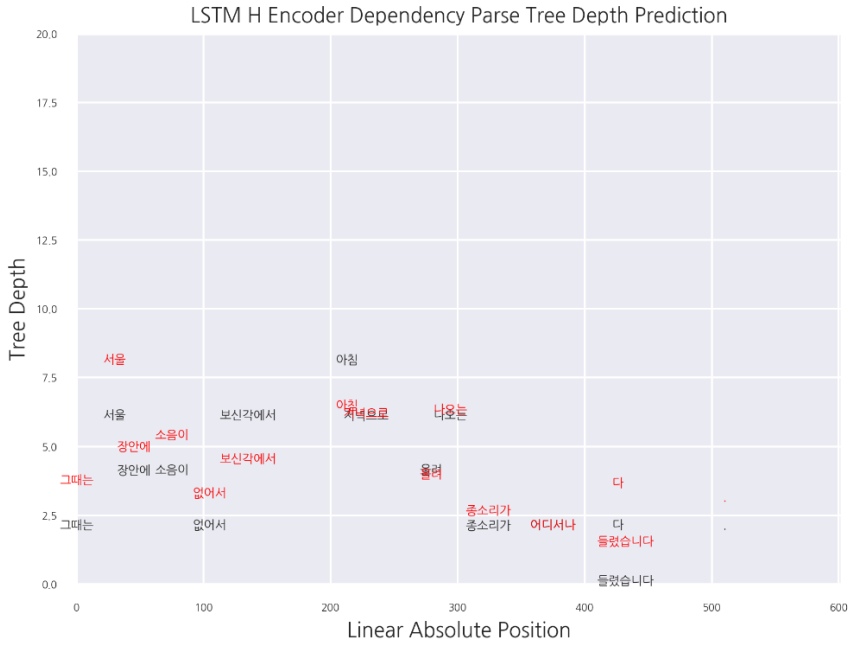


Figure 11. Distance matrix between all pairs of words in a sentence

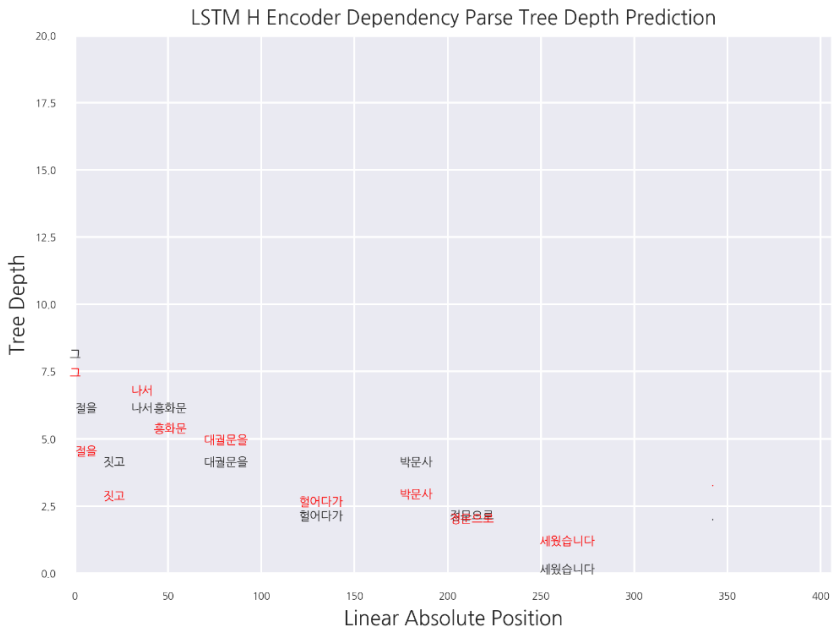
(a)


LSTM H Encoder Dependency Parse Tree Depth Prediction
Tree Depth
Linear Absolute Position
옛날의
서울의
문습이
몹시
몹시
그립습니다
그립습니다

(b)


LSTM H Encoder Dependency Parse Tree Depth Prediction
Tree Depth
Linear Absolute Position
서울
서울
보신각에서
장안에
소음이
장안에 소음이
보신각에서
그때는
없어서
그때는
없어서
아침
아침
처벽으로
나오는
올려
중소리가
중소리가
어디서나
다
다
들렸습니다
들렸습니다

(c)



(d)

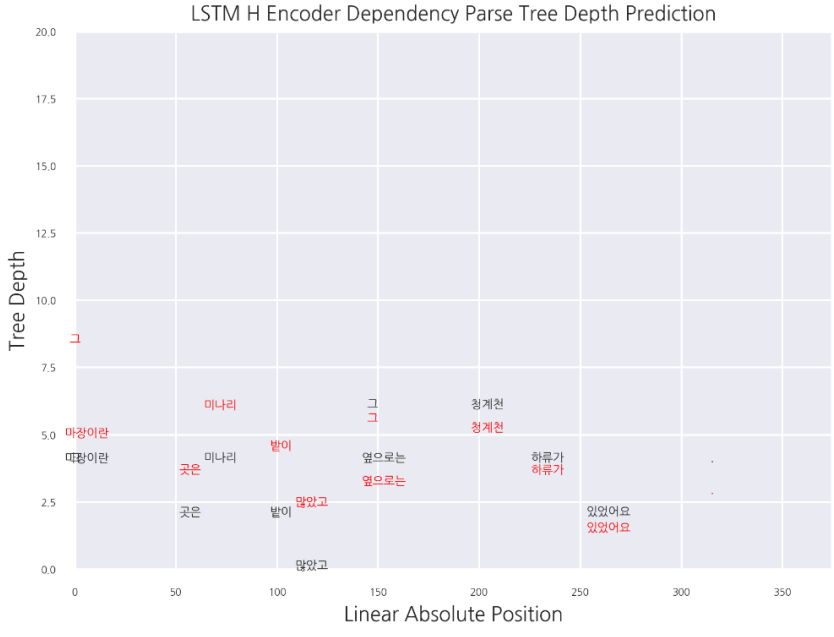


Figure 12. The gold parse trees depth (black) and the predicted norm probes (squared) on maskPOS (red)

6.2. Further Analysis

In order to improve the POS models' performance at the level of the other models with other state-of-the-art models, we investigate further analysis based on the limitation mentioned in the previous chapter. KorNLI (Korean Natural Language Inference) (Ham et al., 2020) is used as the downstream task to compare the performance of the models. The linguistic ability of a model is determined by the performance on the KorNLI task here.

6.2.1. POS Tag Combination

We tried to change the combination of the MeCab POS tag. Other than using a full POS tag combination or the last POS tag, only the first POS tag or the first and the last POS tag combination was implemented. The full POS combinations are suggested in Figure 13.

After pre-training the models to 2M steps (batch size of 16, vocabulary size of 30k) we compared the KorNLI performance of each model. It appears in Table 7 that the full POS tag combination shows the highest performance on the task compared to the other POS tag combinations, especially compared to the combinations with only one tag, the first or the last POS tag. If we consider the first POS tag as lexical information and the other POS tags syntactic information, it can be interpreted that both lexical and syntactic information should be provided for a POS model to learn a latent linguistic representation.

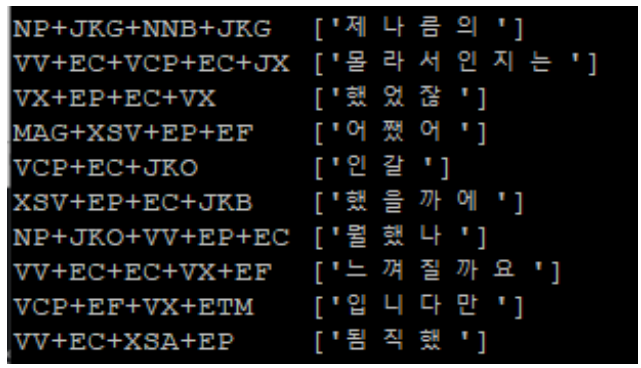


Figure 13. Full POS combinations and example tokens

Model	POS Tag	2M step	KorNLI
maskPOS Batch size = 16 30k vocab	full	(POS) 0.8703 (token) 0.7182	**75.5**
	front+end	(POS) 0.8836 (token) 0.7379	75
	front	(POS) 0.8676 (token) 0.7074	74.97
	end	(POS) 0.8816 (token) 0.7256	74.88

Table 7. Pre-training and fine-tuning results of maskPOS models with different POS tag combinations

6.2.2. Vocabulary Size

We increased the vocabulary size to improve the POS models' ability on the downstream task. Out of 1M tokens from the corpus, 30k, 50k, and 100k vocabulary were chosen in order of frequency and pre-trained to 2M steps (full POS tag, batch size of 16). It turns out that the vocabulary size of

30k that we used in the experiment (chapter 5) is not enough to perform well on the downstream task. As the vocabulary increases, the accuracy of the KorNLI task enhances in Table 8. The maskPOS with a vocabulary size of 100k shows the highest accuracy. Since the POS models implement POS tagger as a tokenizer which is a supervised tokenizer, it must require a bigger vocabulary size compared to the models with WordPiece tokenizer (Wu et al., 2016) which is an unsupervised tokenizer constructing vocabulary based on the statistical frequency of tokens rather than the meaning of them. The result in Table 8 implies that there are possible avenues for improving the POS models.

Model	Vocabulary Size	2M step	KorNLI
maskPOS fulltag Batch size =16	30k	(POS) 0.8703 (token) 0.7182	75.5
	50k	(POS) 0.8821 (token) 0.7237	75.6
	100k	(POS) 0.8908 (token) 0.7310	**76.42**

Table 8. Pre-training and fine-tuning results of maskPOS models with different vocabulary sizes

6.2.3. POS Tagging

In order to avoid POS models from tagging wrong POS tags in the embeddings during pre-training, we construct the off-the-shelf POS tagging model. The off-the-shelf POS tagging model tags POS to a token during the

tokenization process. By tagging POS differently rather than calling POS tags from a pickled dictionary file to avoid wrong tagging, we observe higher performance on the downstream task in Table 9. Although the performance on the KorNLI of the off-the-shelf POS model with a vocabulary size of 30k is lower than that of the original pickle model with the same vocabulary size, the performance of the off-the-shelf model gets better with a bigger vocabulary size. The result reconfirms that the 30k vocabulary size was too small for the POS models to learn enough linguistic representations.

Model	POS Tagging	2M step	KorNLI
maskPOS fulltag Batch size =16	pickle	(POS) 0.8703 (token) 0.7182	75.5
	Off-the-shelf	(POS) 0.8560 (token) 0.6932	74.73
	Off-the-shelf (w/ 100k vocab)	(POS) 0.8999 (token) 0.7370	**75.8**

Table 9. Pre-training and fine-tuning results of maskPOS models with different POS tagging methods

The off-the-shelf POS model would benefit from the lexical information, the first POS tag, as it can disambiguate the word sense. For example, if we implement the off-the-shelf tagging method rather than using a dictionary saved into a pickle file to call a POS tag, tokens in Table 10 which

have the same form with different POS tags can be distinguished.

Token	Frequency	'POS': Frequency
##고	6230238	'EC': 5051649, 'JKQ': 1152139, 'VCP+EC': 18285, 'NNG': 5233, 'XPN': 1246, 'MM': 633, 'XSV+EC': 131, 'VV': 91, 'NNP': 62, 'IC': 21, 'JC': 18, 'NP': 6, 'VA': 4
##과	2518551	'JC': 2093857, 'JKB': 383815, 'NNG': 48575, 'ETN+JKB': 64, 'VV+EC': 61, 'NNBC': 2
전	695925	'NNG': 578685, 'MM': 108186, 'NNP': 2017, 'NP+JX': 654
제	350186	'XPN': 215112, 'NP': 70060, 'NP+JKG': 63324, 'MM': 832, 'NP+VCP': 38, 'XR': 3
사	101443	'NR': 44041, 'VV': 41047, 'NNG': 10752, 'VV+EC': 5474, 'VV+EF': 46, 'NNP': 42
##대로	58585	'JX': 47240, 'NNB': 4979, 'JKB': 4814, 'ETM+NNB': 1191, 'NNG': 287
제한	40979	'NNG': 39174, 'VV+ETM': 1806
지내	17888	'VV': 16762, 'VV+EC': 1031, 'NNG': 59, 'VV+EF': 36
얼	2747	'VV': 1301, 'NNG': 690, 'IC': 471, 'NNP': 210, 'VV+ETM': 76
후진	1743	'NNG': 1564, 'VA+ETM': 179

Table 10. Tokens with different POS tags

A token with high-frequency ##고 *##go* seems to have various POS

tags in Table 9. POS tags are primarily reflected in the lexical information. It mostly appears in conjunctions (EC) or quotations (JKQ). ##과 *##gwa* and ##대로 *##daylo* can be a particle or a part of a noun. Tokens such as 사 *sa,* 제한 *ceyhan,* 지내 *cinay,* 얼 *el,* and 후진 *hwucin* can be nouns or stems of verb/adjective. Some syntactic information reveals in the other parts of POS tags. 전 *cen* and 제 *cey* can appear in nouns or as pronouns with a particle as the first POS tag is NP. They are both 1st person pronouns, but their syntactic information POS tags are different, showing that they have different Cases (+JX/+JKG). A token 지내 *cinay* can be either a stem of a verb (VV) or a verb phrase in itself (VV+EC).

We conclude that the improvement of the off-the-shelf POS tagging model belongs to the ability to use both lexical and syntactic information POS tags.

7. Conclusion

In this study, we incorporated explicit linguistic knowledge into the pre-trained BERT model. Suggesting new training methods by fusing morphosyntactic information, POS tag, with input embeddings, we proposed addPOS, multiaddPOS, and maskPOS models from pre-training. The downstream task results, especially the performances of the maskPOS model, proved the effectiveness of adding linguistic features compared to the base MeCab models. Through the following linguistic probing task and analysis, we insist that the POS models learn a latent linguistic representation during training, and have a potential for future improvement.

One of the limitations in this study is that the performance of the model entirely relies on the quality of a POS tagger. In this case, the ability of a model can be influenced by tagging errors. Also, the 'out-of-vocabulary' (OOV) problem, that the POS tagger cannot infer novel words, must be solved. The preprocessing of raw data required for the pre-training is also a key element in improving the performance of the language model.

It is still under discussion whether the BERT model needs linguistic knowledge for solving its tasks. Glavaš and Vulić (2020) suggested an issue that either BERT has incomplete syntactic knowledge or it does not rely on linguistic information. But obviously, there is a point that BERT understands the structure of a language. Warstadt et al. (2019) studied negative polarity items (NPIs) and found BERT detected the presence and the structure of NPIs (detecting "ever" and the usage of "whether") fairly well, while it was weak to detect scope violations.

We leave for future work language models with more linguistic

features other than POS tags or various architectures as the models in chapter 2. Multi-grained tokenization (Zhang and Li, 2020) or embeddings using Siamese network (Reimers and Gurevyych, 2019) could be applied for future work. Future research could experiment on a more in-depth analysis of the fine-tuning tasks. Various linguistic probing tasks for the Korean language could be developed for future research.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., & Goldberg, Y. (2016). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. arXiv preprint arXiv:1608.04207.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135-146.

Chun, J., Han, N. R., Hwang, J. D., & Choi, J. D. (2018, May). Building universal dependency treebanks in Korean. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does bert look at? an analysis of bert's attention. arXiv preprint arXiv:1906.04341.

Coenen, A., Reif, E., Yuan, A., Kim, B., Pearce, A., Viégas, F., & Wattenberg, M. (2019). Visualizing and measuring the geometry of bert. arXiv preprint arXiv:1906.02715.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., & Baroni, M. (2018). What you can cram into a single vector: Probing sentence embeddings for linguistic properties. arXiv preprint arXiv:1805.01070.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Giulianelli, M., Harding, J., Mohnert, F., Hupkes, D., & Zuidema, W. (2018). Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. arXiv preprint arXiv:1808.08079.

Glavaš, G., & Vulić, I. (2020). Is supervised syntactic parsing beneficial for language understanding? an empirical investigation. arXiv preprint arXiv:2008.06788.

Goldberg, Y. (2019). Assessing BERT's syntactic abilities. arXiv preprint arXiv:1901.05287.

Ham, J., Choe, Y. J., Park, K., Choi, I., & Soh, H. (2020). KorNLI and KorSTS: New Benchmark Datasets for Korean Natural Language Understanding. arXiv preprint arXiv:2004.03289.

Hewitt, J., & Manning, C. D. (2019, June). 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) (pp. 4129-4138).

Hoover, B., Strobelt, H., & Gehrmann, S. (2019). exbert: A visual analysis tool to explore learned representations in transformers models. arXiv preprint arXiv:1910.05276.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., & Levy, O. (2020). Spanbert: Improving pre-training by representing and predicting spans. Transactions of the Association for Computational

Linguistics, 8, 64-77.

Jurafsky, D., & Martin, J. H. (2019). Speech and language processing (3rd ed. draft). Draft of September 16, 2019.

Kovaleva, O., Romanov, A., Rogers, A., & Rumshisky, A. (2019). Revealing the dark secrets of BERT. arXiv preprint arXiv:1908.08593.

Lee, S., Jang, H., Baik, Y., Park, S., & Shin, H. (2020). KR-BERT: A Small-Scale Korean-Specific Language Model. arXiv preprint arXiv:2008.03979.

Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., & Wang, P. (2020). K-BERT: Enabling Language Representation with Knowledge Graph. In AAAI (pp. 2901-2908).

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26, 3111-3119.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013b). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., &

Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in bertology: What we know about how bert works. arXiv preprint arXiv:2002.12327.

Ross, J. R. (1967). Constraints on variables in syntax.

Strubell, E., Verga, P., Andor, D., Weiss, D., & McCallum, A. (2018). Linguistically-informed self-attention for semantic role labeling. arXiv preprint arXiv:1804.08199.

Sundararaman, D., Subramanian, V., Wang, G., Si, S., Shen, D., Wang, D., & Carin, L. (2019). Syntax-Infused Transformer and BERT models for Machine Translation and Natural Language Understanding. arXiv preprint arXiv:1911.06156.

Tenney, I., Das, D., & Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. arXiv preprint arXiv:1905.05950.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Vig, J. (2019). A multiscale visualization of attention in the transformer model. arXiv preprint arXiv:1906.05714.

Wang, Y. S., Lee, H. Y., & Chen, Y. N. (2019). Tree transformer: Integrating tree structures into self-attention. arXiv preprint arXiv:1909.06639.

Warstadt, A., Cao, Y., Grosu, I., Peng, W., Blix, H., Nie, Y., ... & Wang, S. F. (2019). Investigating BERT's Knowledge of Language: Five Analysis Methods with NPIs. arXiv preprint arXiv:1909.02597.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Klingner, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Zhang, X., & Li, H. (2020). AMBERT: A Pre-trained Language Model with Multi-Grained Tokenization. arXiv preprint arXiv:2008.11869.

Zhang, Z., Wu, Y., Zhao, H., Li, Z., Zhang, S., Zhou, X., & Zhou, X. (2019). Semantics-aware bert for language understanding. arXiv preprint arXiv:1909.02209.

Zhou, J., Zhang, Z., Zhao, H., & Zhang, S.(2019). LIMIT-BERT: Linguistic informed multi-task bert. arXiv preprint arXiv:1910.14296.

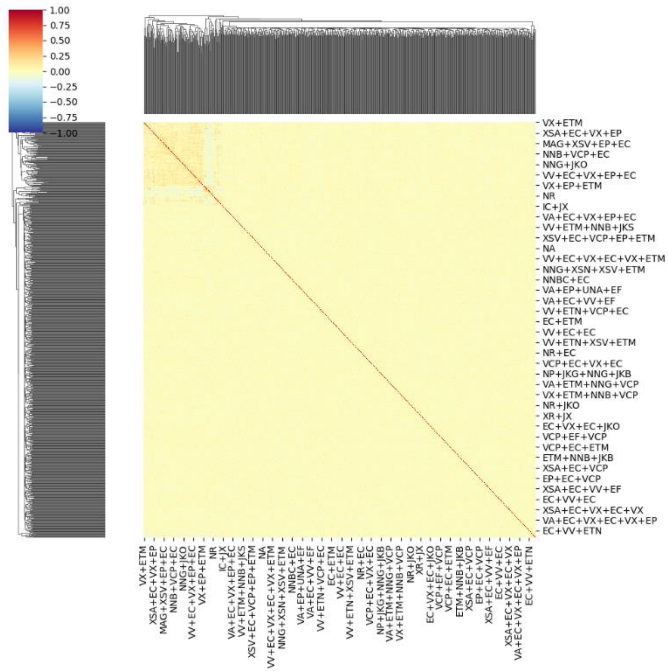
Appendix A. MeCab POS Tag

	대분류(5 언 + 기타)	mecab-ko-dic 품사 태그	
		태그	설명
실질형태소	체언	NNG	일반 명사
		NNP	고유 명사
		NNB	의존 명사
		NNBC	단위를 나타내는 명사
		NR	수사
		NP	대명사
	용언	VV	동사
		VA	형용사
		VX	보조 용언
		VCP	긍정 지정사
		VCN	부정 지정사
	수식언	MM	관형사
		MAG	일반 부사
		MAJ	접속 부사
	독립언	IC	감탄사
형식형태소	관계언	JKS	주격 조사
		JKC	보격 조사
		JKG	관형격 조사
		JKO	목적격 조사
		JKB	부사격 조사
		JKV	호격 조사
		JKQ	인용격 조사
		JX	보조사
		JC	접속 조사
	선어말 어미	EP	선어말 어미
	어말 어미	EF	종결 어미
		EC	연결 어미
		ETN	명사형 전성 어미
		ETM	관형형 전성 어미
	접두사	XPN	체언 접두사

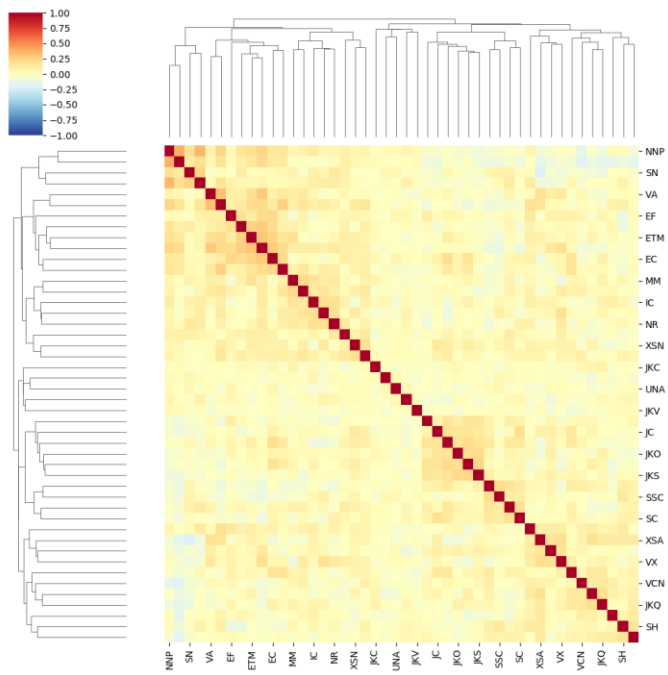
	접미사	XSN	명사 파생 접미사
		XSV	동사 파생 접미사
		XSA	형용사 파생 접미사
	어근	XR	어근
	부호	SF	마침표, 물음표, 느낌표
		SE	줄임표 …
		SSO	여는 괄호 (, [
		SSC	닫는 괄호),]
		SC	구분자 , · / :
		SY	
	한글 이외	SL	외국어
		SH	한자
		SN	숫자

Appendix B. Correlation Heatmap

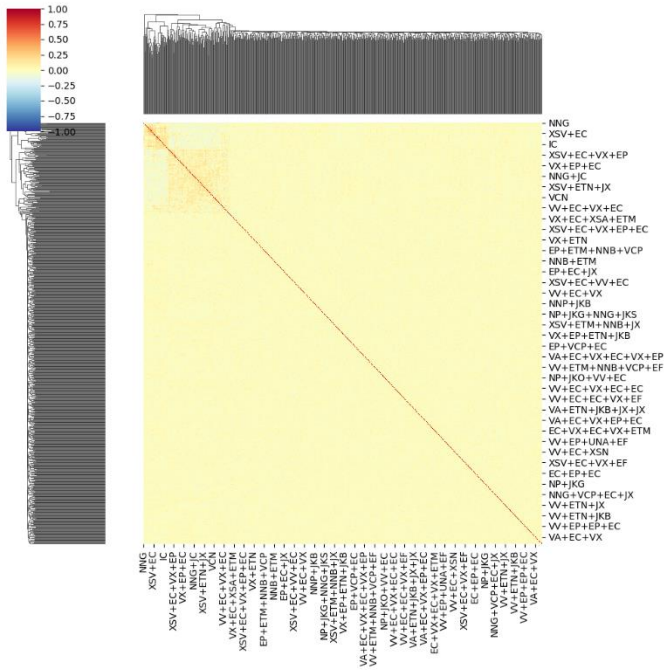
addPOS (fulltag)



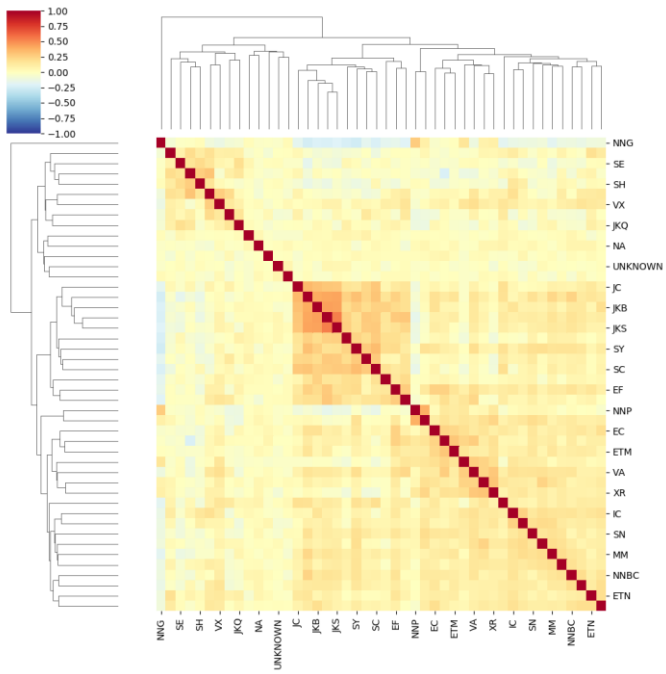
addPOS (endtag)



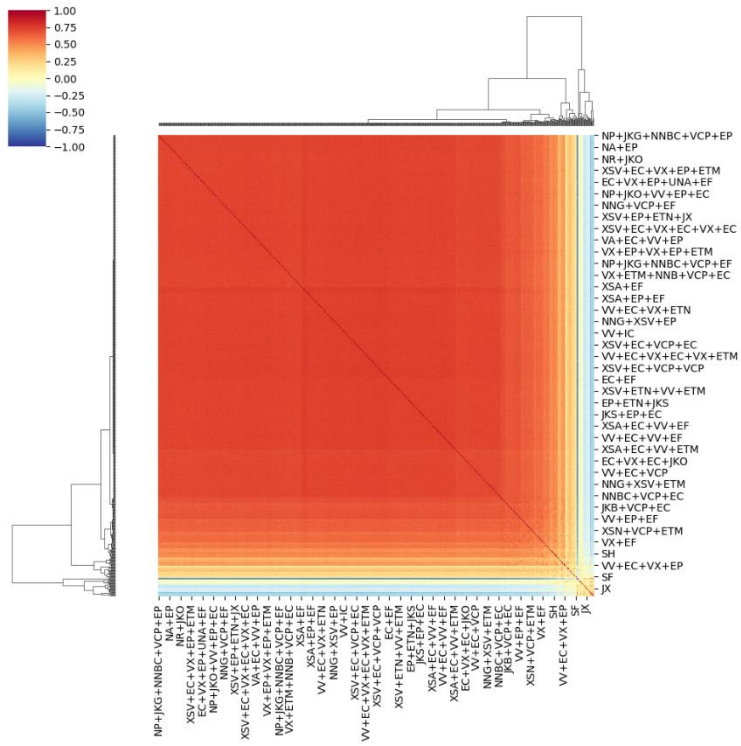
multiaddPOS (fulltag)



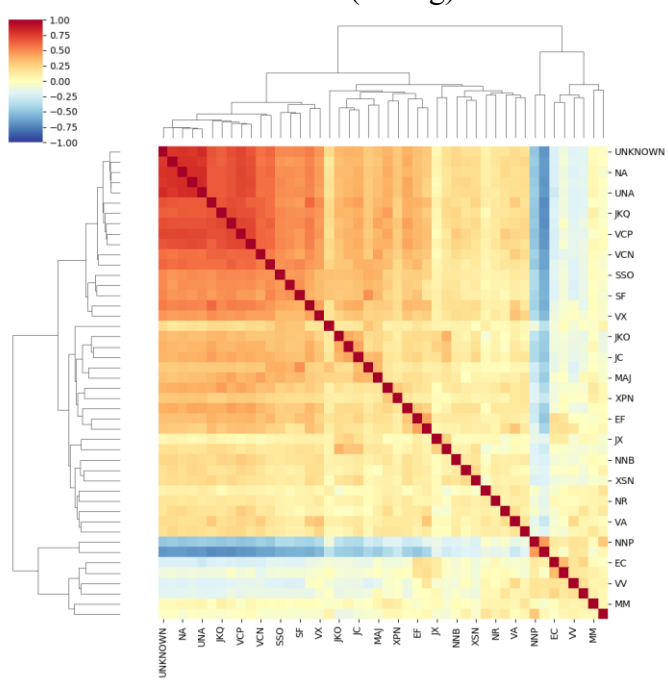
multiaddPOS (endtag)



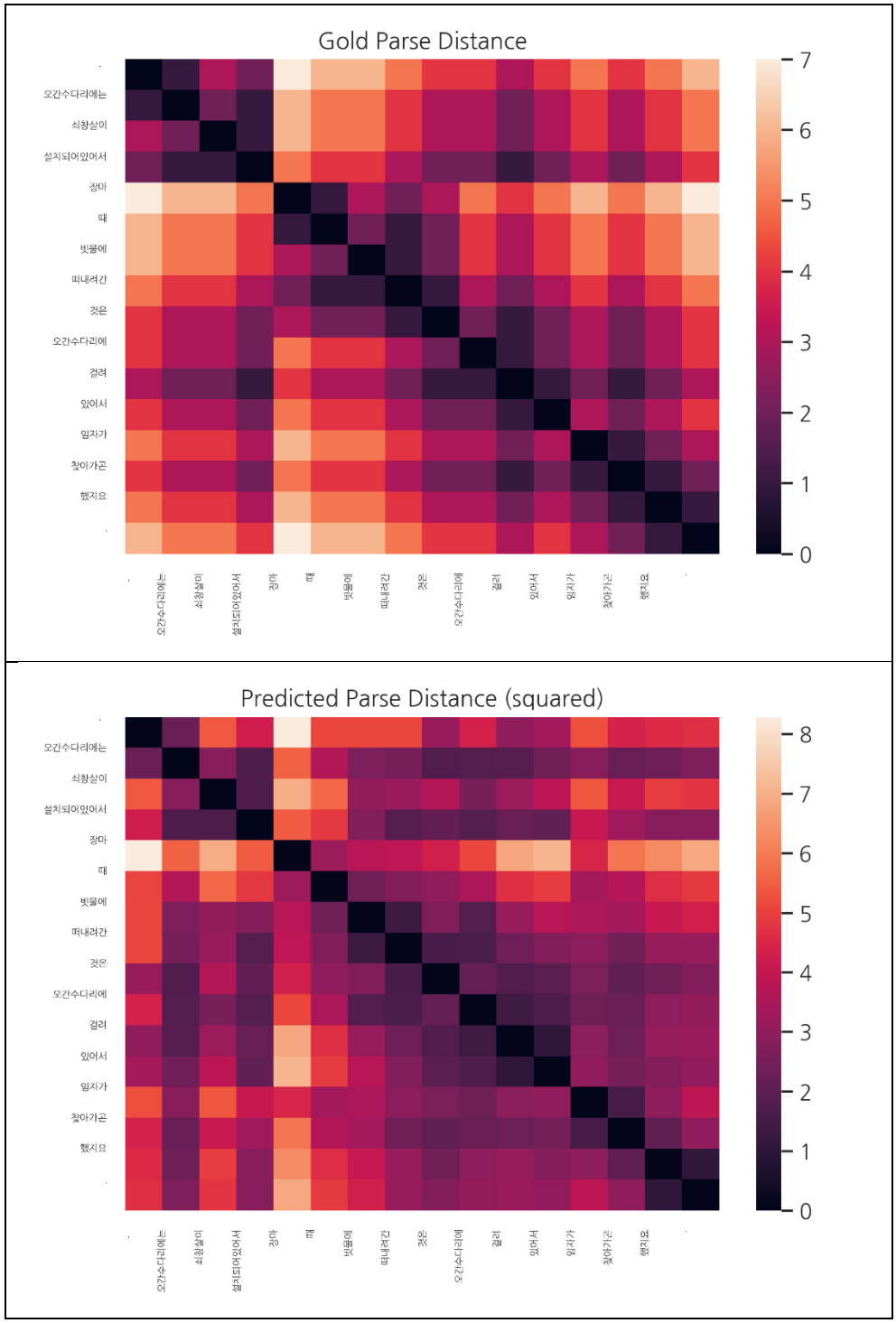
maskPOS (fulltag)



maskPOS (endtag)



Appendix C. Distance Matrix



국문 초록

품사 임베딩 정보를 결합한 언어학적 BERT 모델

백연미

언어학과

서울대학교 대학원

본 연구에서는 BERT 모델에 품사라는 언어학적 정보를 결합하여 모델의 성능을 높이고 이를 언어학적으로 분석하고자 하였다. BERT는 그 자체로 강력한 성능을 내는 모델이지만 모델에 명시적으로 언어학적 정보를 결합하여 주입했을 때 그 성능이 더욱 올라갈 수 있는 여지가 있다는 연구가 이루어지고 있다. 또한 최근 언어 모델이 어떠한 언어학적 지식을 학습했는지 분석하는 연구가 활발하게 이루어지고 있으나 한국어를 대상으로는 사전학습된 모델의 언어학적 표상을 해석하는 분류기(probing classifier) 연구가 아직 미비한 상황이다.

실험을 위해 본 연구에서는 사전학습 단계에서 다양한 방법으로 기존 BERT 모델의 입력 임베딩에 품사 임베딩 정보를 추가하였다. 이에는 (1) 품사 임베딩을 더하는 방법(addPOS), (2) 품사 임베딩을 곱하고 더하는 방법(multiaddPOS), 그리고 (3) 품사 임베딩을 마스킹하는 방법(maskPOS)이 사용되었다. 사전학습 말뭉치로는 한국어 위키피디아와 뉴스기사가 사용되었고 이때 품사는 MeCab 형태소 분석기를 이용하여 태깅되었으며 이는 모델이 말뭉치를 토큰화하는 토큰의 단위로 사용되기도 했다. 이후 학습된 모델을 이용하여 5개의 한국어 하위 실험(downstream task)을 진행하였다(NSMC, NER, KorQuaD, KorNLI, KorSTS). 실험 결과 품사를 명시적으로 결합한 모델, 그 중에서도 maskPOS 모델이 품사 정보가

제공되지 않은 모델보다 높은 성능을 보였다. 하지만 최신 모델에 비해서는 낮은 결과를 내었다.

이후 품사 임베딩 정보가 결합되어 학습된 모델을 대상으로 언어학적 분석을 진행하였다. 모델이 학습한 통사 정보를 확인하기 위해 Hewitt and Manning (2019)에서 제안된 structural probe를 한국어 데이터셋에 적용하여 실험이 이루어졌다. 그 결과 품사 임베딩을 결합하여 명시적으로 언어학적 정보를 준 모델이 한국어 통사 정보를 학습했다는 사실을 확인할 수 있었다. 추가로 품사 모델의 성능을 더 높이기 위해 추가 실험을 진행하였고 품사 모델의 성능을 높일 수 있는 여지가 있다는 결론을 낼 수 있었다.

본 연구는 한국어를 대상으로 BERT 사전학습 모델에 언어학적 정보를 명시적으로 결합하는 새로운 방법을 제시한다. 또한 한국어 모델로는 최초로 모델의 언어학적 표상을 해석하는 연구(probe)를 적용했다. 마지막으로 본 연구는 컴퓨터 공학의 딥러닝 기법과 언어학 이론을 결합하며 앞으로 한국어 자연언어처리가 나아가야 할 방향을 제시한다.

주요어: 자연언어처리, 언어 모델, BERT, 임베딩, 품사, 모델 해석, Probe, 파스 트리

학번: 2018-20037