Received: 23 September 2019 | Revised: 14 February 2020 | Accepted: 6 May 2020

DOI: 10.4218/etrij.2019-0441

ORIGINAL ARTICLE

ETRI Journal WILEY

Zero-anaphora resolution in Korean based on deep language representation model: BERT

Youngtae Kim[1] | **Dongyul Ra**[1] | **Soojong Lim**[2]

[1]Computer and Telecommunications Engineering Division, Yonsei University, Wonju, Rep. of Korea

[2]Language Intelligence Research Section, Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea

Correspondence
Dongyul Ra, Computer and Telecommunications Engineering Division, Yonsei University, Wonju, Rep. of Korea.
Email: dyra2246@gmail.com

Funding Information
This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Rep. of Korea (2017R1D1A3B03031855), and by Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT), Rep. of Korea (No. 2013-0-00131, Development of Knowledge Evolutionary WiseQA Platform Technology for Human Knowledge Augmented Services).

Abstract

It is necessary to achieve high performance in the task of zero anaphora resolution (ZAR) for completely understanding the texts in Korean, Japanese, Chinese, and various other languages. Deep-learning-based models are being employed for building ZAR systems, owing to the success of deep learning in the recent years. However, the objective of building a high-quality ZAR system is far from being achieved even using these models. To enhance the current ZAR techniques, we fine-tuned a pre-trained bidirectional encoder representations from transformers (BERT). Notably, BERT is a general language representation model that enables systems to utilize deep bidirectional contextual information in a natural language text. It extensively exploits the attention mechanism based upon the sequence-transduction model Transformer. In our model, classification is simultaneously performed for all the words in the input word sequence to decide whether each word can be an antecedent. We seek end-to-end learning by disallowing any use of hand-crafted or dependency-parsing features. Experimental results show that compared with other models, our approach can significantly improve the performance of ZAR.

KEYWORDS

attention, bidirectional encoder representations from transformers (BERT), deep learning, language representation model, zero-anaphora resolution (ZAR)

1 | INTRODUCTION

Zero anaphora refers to a phenomenon in which pronouns to fill obligatory grammatical roles such as subject and object of predicates are omitted in natural-language texts. It occurs frequently in some languages such as Japanese, Korean, Chinese, and Italian. The omitted pronoun is called the *zero pronoun* (ZP). The predicate at which a ZP occurs is called the ZP predicate. It is assumed that the location of the ZP is the same as that of its predicate. Subjects are omitted significantly more frequently than objects. We confine our zero-anaphora resolution (ZAR) system to deal with only ZPs attributed to the omission of a subject.

In a document that contains a ZP, there may exist one or more noun phrases (NPs) that co-refer to the same entity in the world as the ZP. These NPs are referred to as *antecedents* of the ZP. Antecedents occur before the ZP. Therefore, the NPs that appear before ZP are called candidate antecedents (CAs).

Given a document, we must detect ZPs and find their antecedents. This process is called *ZAR*. For better understanding of texts, we must achieve high performance in ZAR. Assume there is a ZP in a text. If there exists at least one

antecedent, then ZP is said to be anaphoric. Otherwise, it is said to be non-anaphoric. Both the cases frequently occur in texts.

In Figure 1, the subject of a verb 떠났다 is omitted, resulting in a ZP. The symbol ϕ indicates the ZP. The properties of this ZP are as follows:

ZP predicate: 떠났다;

CAs: "그 소녀는", "도서관에", "많은 학생들이", "책을", "어머니로부터", "전화가", "집으로";

Antecedents: "그 소녀는".

We require our ZAR system to find the main word of an NP that is an antecedent. Our system need not recognize the entire NP. More details will be provided in Section 4.1. In Figure 1, the word 소녀는 is the main word of the antecedent NP "그 소녀는. Among the words in an NP, the last word is the main one in Korean.

Generally, a ZP can have zero or more antecedents. ZAR involves two subtasks: (i) detecting a ZP, and (ii) searching the antecedent of the ZP [1–3]. The detection subtask is required because there exist no explicit marks in a text, indicating the occurrences of ZPs. After the antecedent search is completed, the system can decide whether the ZP is non-anaphoric or anaphoric. This decision regarding anaphoricity can be made using the result of the antecedent-search process. The configuration of our ZAR system is depicted in Figure 2.

Devlin and others [4] recently introduced a new language representation model called BERT, which employs "Transformer," a deep-learning architecture that extensively exploits attention mechanism [5]. Transformer can model the encoding and decoding processes of language-understanding tasks. BERT is a general language representation model based on Transformer. It is pre-trained using significant amount of unlabeled corpora. It can model deep bidirectional contexts of words to the left and right of a text. It can also be used as an underlying language model for various language-understanding tasks. It was shown that the approach of fine-tuning BERT allows the development of state-of-the-art systems in many tasks.

ZAR is a difficult task. The performances of best systems up to now are not high even after a long history of research. It is interesting to see whether finetuning a pre-trained BERT can enable systems to achieve a significant improvement in ZAR. In this study, we will describe our work done to answer this question. Among the two subtasks of ZAR in Figure 2, antecedent search has received more attention in research than ZP detection. ZP detection can rely on the syntactic analysis of sentences or binary classification of predicates. Antecedent search seems to require more intelligence than ZP detection. Additionally, the possibility of a ZP being non-anaphoric makes the search task difficult. Similar to many other recent research works on ZAR, we focus on antecedent search.

그 소녀는	도서관에	갔다.	많은	학생들이	책을	읽고	있었다.
(the girl	to a library	went.	many	students	books	reading	were.)
나중에	어머니로부터	전화가	왔다.	서둘러	집으로	ϕ떠났다.	
(later	mother	phone-call	arrived.	in a hurry	home	left	)

FIGURE 1 Example of a ZP in a Korean text.

We developed an antecedent-search module by fine tuning a pre-trained BERT. This search module is based on a deep-learning model, which is constructed by adding a neural-network architecture on top of BERT. Fine-tuning is performed using a Korean tagged corpus annotated for ZAR.

We also developed another antecedent-search module based on various other machine-learning models, such as the structural support vector machine (S-SVM) [6] and pointer network [7,8]. Our BERT-based model outperformed these aforementioned models.

We also performed experimentations to compare our technique with those in recent research works based on various deep-learning models [9–11]. We developed antecedent-search modules based on these models. The same Korean tagged corpus was used in developing these models. The experiment shows that our model is superior to their models in terms of ZAR performance.

This paper is organized as follows. The related research is explained in Section 2. A concise introduction to Transformer and BERT is presented in Section 3. In Section 4, we describe how our search module that employs a BERT-based model is constructed. Experimental results and discussions are provided in Section 5. Conclusions are drawn in Section 6.

2 | RELATED WORK

Many studies have been performed on ZAR in the past. The research witnessed four major trends, which mostly appeared consecutively. First, most research works utilized rule-based frameworks [2,12–14]. As corpus linguistics became active, statistical or probabilistic models were exploited [1,15,16]. Subsequently, as machine learning gained relevance, various machine-learning models were used for developing ZAR systems [3,17,18]. As deep-learning technology has recently flourished in the domain of artificial intelligence, deep-learning models have also begun to be utilized in ZAR [9,10,11,19].

Chen and Ng [9] developed a system based on deep neural networks for ZAR in Chinese. They proposed a method for developing an antecedent-search module when an anaphoric ZP is given. However, one limitation of their work was that they did not suggest a way to distinguish between anaphoric and non-anaphoric ZPs. Their model employed a simple

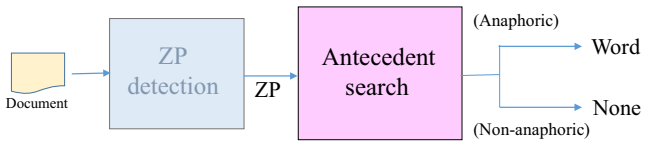

FIGURE 2 Configuration of our ZAR system.

architecture that comprised a feed-forward neural network (FFNN). Cosine similarity was adopted to select an antecedent out of CAs at a late stage of the system, thereby deviating from the deep-learning approach.

Yin and others [10] attempted to improve the antecedent-search model of Chen and Ng [9]. Their model also required the input to be an anaphoric ZP. However, they extensively used attention mechanism, inspired by the recent trend of deep learning. Self-attention was used for obtaining a representation vector of the ZP. A recurrent neural network (RNN) that used long short-term memory (LSTM) blocks was utilized to encode the context around the ZP. A NP chosen as a CA was also modeled using an LSTM-RNN. An attention mechanism was used to produce a vector that represented the NP. These two vectors, along with a vector that represented handcrafted features, were used as the input to an FFNN. Subsequently, the FFNN outputs the probability that the CA is an antecedent of the ZP. Their advanced architecture improved the performance by approximately 3% relative to the model of Chen and Ng [9]. However, ZP detection and anaphoricity decision were beyond the scope of their work.

Iida and others [11] proposed a method for Japanese ZAR based on multi-column convolutional neural networks (MCNNs). An NP (as a CA) and a ZP predicate are fed to the MCNN model as a part of the input. The output of MCNN is the probability that the CA is an antecedent of the ZP. The MCNN comprises multiple convolutional neural networks (CNNs). A text segment is given to each CNN, which then produces a vector that represents the input. Subsequently, the output vectors of all the CNNs are concatenated and inputted to an FFNN, which forms the upper layers of the MCNN. The FFNN outputs the probability that the CA is an antecedent of the ZP. However, a limitation of their work is that only a single sentence that contains the ZP constitutes the search space. ZP detection was not discussed in their work as well.

Jung and Lee [19] used an advanced deep neural architecture for recovering dropped pronouns in Korean. Given a sentence as the input, their system performed two subtasks: detecting a ZP and determining the type of the ZP (if it exists). However, antecedent search is not performed by their system. Therefore, the goal of their work is different than ours.

After observing that BERT could be used to develop state-of-the-art systems for various language tasks, new language representation models with a similar purpose have been proposed, and these models possess several advantages [20–22]. Exploiting these new models may enable us to further enhance our ZAR performance. Yin and others [23] added collaborative filtering to their original method [10] to improve performance. However, the performance gain over their original method was not significant. Kong and others [24] proposed a chain-to-chain approach in ZAR. In their method, ZP resolution was attempted between ZP co-referential chains and common NP co-referential chains, thereby reducing the search space. Their method achieved an F-score of 55.8 in ZP resolution, and this is the best score among ZAR systems in Chinese.

3 | BERT

Bidirectional encoder representation from transformers (BERT) is a recently introduced general language representation model [4]. The linguistic motivation behind using BERT can be stated as follows: BERT can encode deep bidirectional contextual information both on the word and sentence levels; it can encode phrase-level information in lower layers; it can encode a rich hierarchy of linguistic information such as surface features at bottom layers, syntactic features in the middle layers, and semantic features at the top layers; it can encode long-distance dependency information on deeper layers; it can capture linguistic information in a compositional manner similar to that employed in classical tree-like structures.

A pre-trained BERT can be exploited to develop systems for various language tasks. The pre-training approach is effective in building state-of-the-art models.

3.1 | Transformer

For sequence-modeling and transduction problems, such as language modeling and machine translation, the RNN architecture has been widely adopted in deep learning [25]. Because of the vanishing gradient problem, it was proposed to replace the units of RNN with LSTM units, which have a set of gates that control the transfer of signals [26]. Gated recurrent units were also introduced with a similar purpose [27]. They were shown to have advantages over simple units of RNN [28]. Attention mechanism was introduced to enhance sequence models based on these RNN-type models [29].

To advance sequence modeling and transduction, a model named Transformer was introduced, which offers advantages in modeling global contextual dependencies [5]. Transformer eschews recurrence for the better exploitation of attention mechanism, allowing the modeling of long-distance dependencies and enabling efficient parallelization. Transformer comprises an encoder and a decoder. Because BERT uses only the encoder part of Transformer, our explanation will

be confined to this part only. Figure 3 depicts an encoder that comprises n layers. In each layer, attention operation is performed, and its result is fed to an FFNN. Each attention step and each feed-forward step are followed by an "add and normalization" step (not shown in Figure 3) [30].

The same architecture is shared across all the positions. Let $x_i^{(l)}$ denote the output vector of layer l at position i. Let H be the size of the output vector of each layer. Let $X^{(l)}$ be a matrix whose ith row is $x_i^{(l)}$. $X^{(0)}$ is initialized by using the word-embedding vectors of the tokens in the input sequence. Transformer specifically uses the dot-product attention implemented by matrix multiplication as follows. Compatibility D is computed as follows:

$$D^{(l)} = X^{(l-1)} X^{(l-1)\mathrm{T}}. \tag{1}$$

Let τ be the length of the input sequence. $D^{(l)}$ is a $\tau \times \tau$ matrix, where $D_{i,k}^{(l)}$ represents the compatibililty between the output of layer $l-1$ at position i and that of layer $l-1$ at position k. A row-wise softmax operation is applied to $D^{(l)}$ to obtain weight matrix $S^{(l)}$ as follows:

$$S_{i,k}^{(l)} = \frac{\exp\left(D_{i,k}^{(l)}\right)}{\sum_{j=0}^{\tau-1} \exp\left(D_{i,j}^{(l)}\right)}, \tag{2}$$

for all $i, k, 0 \le i \le \tau - 1, 0 \le k \le \tau - 1$. For brevity, we write this softmax operation as follows:

$$S^{(l)} = \mathrm{softmax}\left(D^{(l)}\right). \tag{3}$$

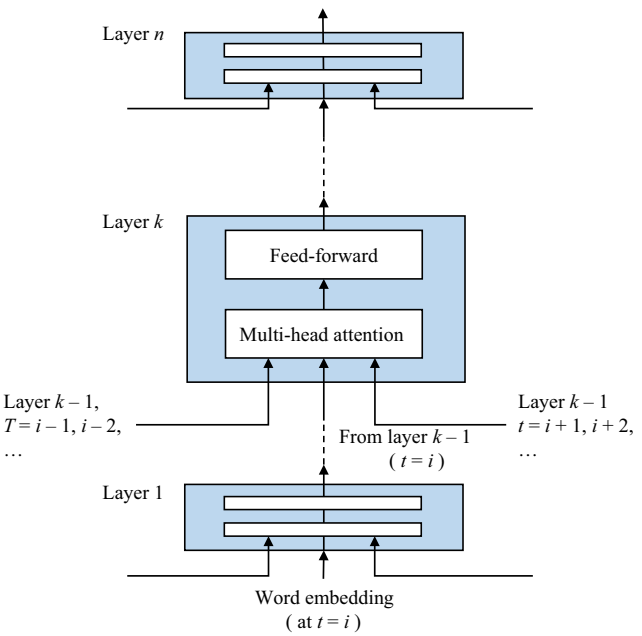

FIGURE 3 Transformer model (encoder part) at position i.

Furthermore, $A^{(l)}$, which is computed using (4), is the result of attention step at layer l for all the positions. Row i of $A^{(l)}$ is the output vector at position i of the attention step of layer l.

$$A^{(l)} = S^{(l)} X^{(l-1)}. \tag{4}$$

Each row i of softmax $S^{(l)}$ is used for weighting the vectors of layer $l-1$ of all the positions to obtain an output vector for the specific position i.

Transformer adopts multi-head attention, which is an extension of the previously introduced basic attention. To compute a head, head_i, $X^{(l-1)}$ is projected to three matrices Q, K, and V as follows, where W_i^Q, W_i^K, W_i^V denote parameter matrices:

$$Q = X^{(l-1)} W_i^Q, \tag{5}$$

$$K = X^{(l-1)} W_i^K, \tag{6}$$

$$V = X^{(l-1)} W_i^V. \tag{7}$$

Subsequently, the ith head is computed as follows:

$$\mathrm{head}_i = \mathrm{softmax}\left(QK^{\mathrm{T}}\right) \mathrm{V}. \tag{8}$$

In the case of multi-head attention with h heads, the final result of attention $A^{(l)}$ is obtained as follows:

$$A^{(l)} = \mathrm{concat}\left(\mathrm{head}_1, \ldots, \mathrm{head}_h\right) W^O, \tag{9}$$

where concat denotes row-wise concatenation. $W^O \in R^{hH \times H}$ is a matrix that projects multi-heads to the attention result. The projection matrices Ws are the parameters of the model, and their values must be learned with training. The FFNN part in each layer involves many weights and biases, which are also the parameters of the model.

3.2 | Pre-training BERT

BERT is a language representation model that adopts the encoder part of Transformer as its architecture [4]. It significantly exploits self-attention and can learn deep bidirectional contexts. It is pre-trained using a large unlabeled corpus.

Figure 4 depicts situations in which BERT is being pre-trained. An input sequence to BERT comprises two special tokens and words of two text segments A and B. The special token [CLS] is placed in front of segment A, and [SEP] is inserted between both the segments. To counter the unknown word problem, the words in the segments are transformed to

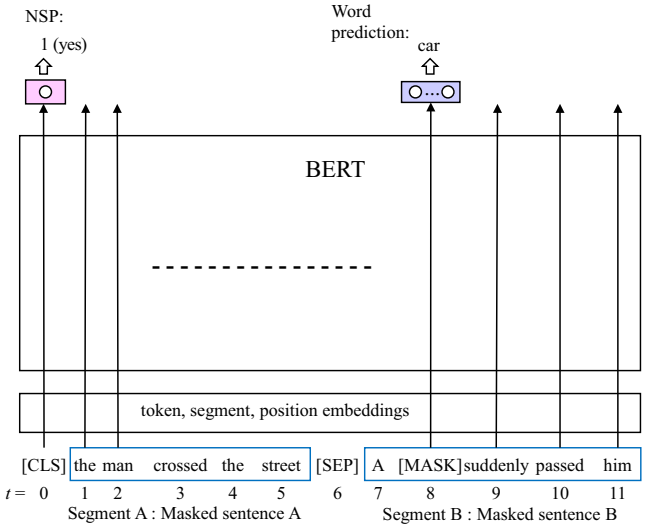

FIGURE 4 Using two tasks for pre-training BERT.

the sequences of word pieces [4,31–33]. Therefore, the input sequence to BERT comprises two special tokens and word-piece tokens. At each position, the input vector to BERT is formed by concatenating three embedding vectors that correspond to token, segment, and position, respectively [4].

BERT is pre-trained using two schemes. The first one is called "masked" language modeling, which enables BERT to be deeply bidirectional. In this scheme, the words in the corpus are randomly selected and masked using string [*MASK*]. The training aims to predict the original words at the positions with [*MASK*]. Via this training scheme, BERT can absorb token-level bidirectional deep context because its self-attention is bidirectional and based on deep Transformer architecture with more than 12 layers. In Figure 4, the input word at position 8 is masked. At the output layer added on the top of BERT, the masked word in the input is predicted. Training is performed with "car" (which was masked) as the target of prediction.

Many applications, such as paraphrase, entailment, and language inference, require sentential-relationship knowledge. To enable BERT to acquire such knowledge, the next-sentence prediction (NSP) task is used for pre-training. Each of segments A and B is fed with a sentence. Accordingly, the NSP is to make a binary decision whether it is natural that the sentence in segment B immediately follows the sentence in segment A in real-world texts. An output layer is added on the top of BERT at position 0 (whose token is [CLS]) to perform binary classification, as depicted in Figure 4.

An unlabeled corpus is used for pre-training. Therefore, it is not difficult to acquire training data of large size. The parameters related to the added layers are trained during the BERT fine-tuning process. The parameters of BERT are also trained in this process.

4 | ZAR WITH BERT

To build a model based on BERT for an application, a neural-network architecture is added on the top of pre-trained BERT. Two approaches were suggested related to the training of the BERT-based model: fine-tuning and feature-based approaches [4]. In the fine-tuning approach, the parameters in both added architecture and BERT are updated via fine-tuning. However, in the feature-based approach, the parameters only in the added part are updated. We employ the fine-tuning approach to build our ZAR model.

4.1 | Annotations used in tagged corpus

A ZAR tagged corpus is used for training and testing in developing ZAR systems. We introduce an annotation scheme used to construct our tagged corpus. This enables us to explain our model more clearly. Given a text document used for tagging, all the ZPs should be identified and manually tagged. This involves tagging ZP predicates and their antecedents. In this study, the terminology "words" represents the strings separated by spaces in a sentence. A word is a unit called "eojeol" in Korean. We adopt a single-word-tagging policy. In our tagging rule, only one word is tagged as a ZP predicate. The same rule applies to the antecedents. If a ZP predicate or an antecedent is a multi-word phrase, only the head word among them is tagged.

An example of annotation is depicted in Figure 5. A ZP predicate is a compound predicate "가지 못했다" that comprises two words. Only the head word 가지 is tagged as the ZP predicate by enclosing it within tags "</p*n*:" and "/>." An antecedent of a ZP is an NP. It may comprise more than one word. In Figure 5, the NP "금발머리의 미국소녀가" is an antecedent of the ZP. Although the NP has two words, only the head word 미국소녀가 is tagged as antecedent by using tags "</a*n*:" and "/>." Notably, the word 금발머리의 of the NP is not annotated. The number *n* in tags "</p*n*:" or "</a*n*:" is used to associate an antecedent to its ZP predicate. Notably, 미국소녀가 contains a compound noun. Because it is a single word, its entire part is annotated.

In the tagged corpus, every ZP predicate constitutes one ZAR problem. A ZP may have zero or more antecedents. In Figure 5, because there is only one ZP predicate, there exists one ZAR problem, which comprises the following components, along with the text of the document:

ZP predicate: "가지 못했다";
Antecedents: "금발머리의 미국소녀가", "그녀는".
The ZAR system should perform the following subtasks:

- ZP detection: This task is to detect ZPs that exist in a document. Among all the predicates in the document, the

predicates whose subjects are missing should be identified. For example, predicate "가지 못했다" in Figure 5 does not have a subject. Therefore, a ZP occurs at this predicate. Notably, 머물렀다 has a subject 그녀는, and thus a ZP does not occur at this predicate.

- Antecedent search: This task is to find the antecedents of a ZP, which is sent by a ZP-detection module. The goal of this task in this study is to find one of the antecedents of the ZP. If the ZP is anaphoric, the system must find one word as an antecedent. Otherwise, the system should notice that there is no antecedent.

ZP detection is simpler than antecedent search and, thus, received less attention by researchers.[1] Antecedent search is sometimes called ZP resolution. We aim to develop a module in charge of antecedent search. Therefore, ZPs annotated in the tagged corpus are used as input to our antecedent-search module for training and testing.

4.2 | Proposed model

We propose to utilize a deep-learning model for developing an antecedent-search module depicted in Figure 2. Accordingly, we fine-tune a pre-trained BERT to develop the module.

The deep-learning model employed by our module is based on BERT, as depicted in Figure 6. As suggested in [4], a neural-network architecture is added on the top of BERT so that the added part can produce an output appropriate for a downstream task, such as ZAR. Our first model, called ZB1, is built by adding only one output layer O, as depicted in Figure 7A. Because the model should perform binary classification for each input token into A (antecedent) and N (not antecedent), the output layer comprises two neural units. The softmax output of the first unit, y_0 represents probability P_A that the input token is classified as A by the model. The output of the second unit, y_1, is the probability P_N that the token is classified as N. Notably, $P_A + P_N = 1$.

Another model ZB2 depicted in Figure 7B has one more layer B between layers O and n. Layers O and B are fully connected FFNN layers. Although it was suggested in [4] that it is sufficient to add only simple FFNN layers as in ZB1 and ZB2, we built another model called ZBL by adding an RNN layer on the top of BERT. We used LSTM units for the RNN. The motivation is that performance improvement can be expected because an RNN can encode the history information.

The input to our model is a sequence of tokens, $X = (x_0, \dots, x_{\tau-1})$, which corresponds to the "tokens" row in Figure 6. We denote the input-sequence length by τ in this paper. Our model performs sequence labeling on the input sequence. Thus, the output is a sequence of labels $Y = (l_0, \dots, l_{\tau-1})$, where l_i denotes a label to which x_i is assigned by the model. The set of all the possible output labels in our model is denoted by $L = \{A, N\}$ as previously mentioned.

[1]For ZP detection, a dependency-parsing result is generally used. Alternatively, a machine-learning-based classifier may be used to detect a predicate whose subject is missing.

```
금발머리의 </a1:미국소녀가/> 한국에 왔다.
</a1:그녀는/> 주로 서울 지역에서 머물렀다.
시간이 부족해서 남부 지방에 </p1:가지/> 못했다.
```

FIGURE 5 Annotation in the tagged corpus.

To construct an input, our model uses two text segments A and B. We feed a word of a ZP predicate into segment A. The words in the text from the beginning of the document until the ZP predicate are placed in Segment B. Figure 6 depicts two segments that comprise words for the ZAR problem of the ZP shown in Figure 1.

Each word in Segments A and B is divided into one or more tokens (or subwords) using a tokenizer. The words here correspond to "eojeols" in Korean. Tokens (more primitive than words) are used as input units to BERT to reduce the unknown word (or out of vocabulary) problem. The word-piece model is used for tokenization in Google BERT [31,32], while byte-pair encoding is used in the BERT specialized for Korean [33].

The input sequence to BERT, $X = (x_0, \dots, x_{\tau-1})$, is formed by using the tokens from words in Segments A and B and two special tokens [CLS] and [SEP]. As depicted in Figure 6, [CLS] is placed at the first position. [SEP] is placed at a position between Segments A and B and at the last position. At each token position, an input vector that corresponds to the token is formed by summing token embedding, segment embedding, and positional embedding, as explained in [4]. The input vectors of all the token positions constitute an input sequence, which is the actual input to the BERT model.

Conceptually, the output of the model can be used to construct a label sequence $Y = (l_0, \dots, l_{\tau-1})$. Our model produces two label probabilities P_A and P_N (from the two units of layer O) at each token position. It works on the token level. The tokens from a same word may have different P_A values. The system (ie, the antecedent-search module in this study), which employs the BERT-based model, should produce an output at the word level. Conversely, a word that can be an antecedent must be recognized and outputted. Therefore, from the token-level output of the model, the word-level output of the system should be obtained, as will be detailed in Section 4.4.

For example, the row P_A in Figure 6 shows the P_A values of tokens for the ZP in Figure 1. This output of the model appears satisfactory because P_A is high for the tokens from

the antecedent word "소녀는" and low for all tokens from non-antecedent words and special tokens.

The Korean morphology is complex. A word in the compound-noun category contains multiple nouns as morphemes. It may have a suffix called "josa" as the last morpheme. However, we consider a compound-noun word similar to a simple noun. Without distinguishing between a compound noun and simple noun, both of them are transformed to tokens by using the tokenizer. Subsequently, the tokens are fed to the input token sequence. Thus, no special treatment is applied only to compound-noun words. We process all the tokens produced from any word in the same way in our model.

In [4], the first token of a word is assigned a different role than the remaining tokens. For example, in the BERT-based model for NER, the first token from a word is assigned a target label *B-PS,* and a special target label "*X*" is assigned to other tokens of the word. For inference, their model used the probability output of the first token to make a decision on the word (outputs from other tokens are ignored). However, all the tokens of a word are utilized in our approach. Our all-token (AT) approach was superior to the first token (FT) only approach in terms of performance. Further details on token processing will be described in Section 4.4.

4.3 | Fine-tuning of BERT

A training example for fine-tuning is prepared for each ZP that exists in the ZAR tagged corpus. It comprises a pair, (input token sequence, target label sequence). Figure 6 depicts an example of a target label sequence that corresponds to an input token sequence for the ZP in Figure 1.

In our model, target output labels that correspond to the input tokens are prepared as follows. All the tokens generated from the words with an antecedent tag in Segment B are assigned target label *A*. Target label *N* is assigned to (a) all the tokens from all other words in Segment B, (b) all the tokens from all the words in Segment A, and (c) two special tokens [CLS] and [SEP]. An example is depicted in Figure 6. Target label *A* is assigned to all the tokens from word 소녀는 in Segment B because word 소녀는 is tagged as an antecedent in Figure 1. Subsequently, all the other tokens in the input sequence are assigned target label *N*.

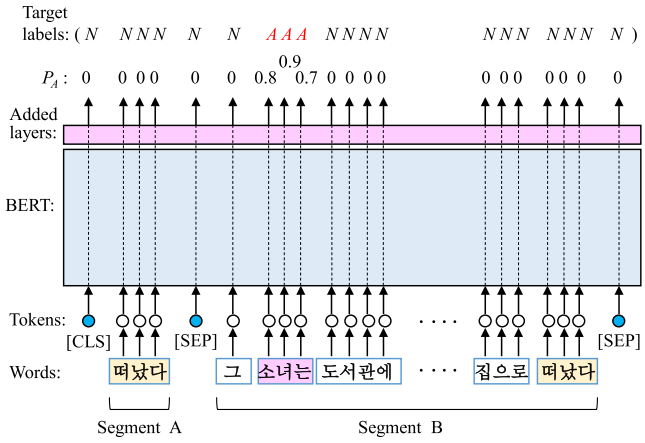

FIGURE 6 Overall configuration of our model based on BERT including input and output.

If the ZP is anaphoric, there exists at least one word that is tagged as an antecedent. Thus, there will be at least one token whose target output label is *A*. However, if the ZP is non-anaphoric, the target labels for all input tokens should be *N*.

Let us explain training only for model ZB1, which is depicted in Figure 7A. Unit 0 of layer *O* corresponds to label *A* and unit 1 to label *N*. Let W^F be the weight matrix between the output layer *O* and layer *n* of BERT. The first row of W^F has the weights of connections that are directed into unit 0 of layer *O* from units of layer *n*. The second row is for unit 1 of layer *O*. Let $X^{(n)}$ be the output matrix of the last layer *n* of BERT. Row *i* of $X^{(n)}$ is the output vector of layer *n* at position *i*. The output of layer *O* for all the positions is computed via matrix multiplication as follows:

$$F = X^{(n)} \mathrm{x} \left(W^F\right)^{\mathrm{T}}, \tag{10}$$

where T denotes transpose operation.

The element (with row index *i* and column index *u*) $F_{i,u}$ of matrix *F* is the value of unit *u* of layer *O* at position *i*. We use τ to denote the input-sequence length. Matrix *Y* is the softmax that corresponds to *F*. $Y_{t,u}$ (an element of row index *t* and column index *u*) is computed as follows:

$$Y_{t,u} = \frac{e^{F_{t,u}}}{e^{F_{t,0}} + e^{F_{t,1}}}, \tag{11}$$

for all *t*, $0 \le t \le \tau - 1$ and all *u*, $0 \le u \le 1$. The element $Y_{t,u}$ is the probability that, at time *t*, the label that corresponds to unit *u* is the label for the input token. Let d_t be the index of the target (gold) label for a token at position *t*. Accordingly, the loss *L* of our model can be defined using cross-entropy as follows:

$$L(\theta) = -\sum_{t=0}^{\tau-1} \ln\left(Y_{t,d_t}\right), \tag{12}$$

where θ denotes the set of all the parameters in our model. Training is performed to calculate the θ that minimizes *L*.

4.4 | Inference

An ideal system must search the entire document to find an antecedent of the given ZP. Because antecedents can occur

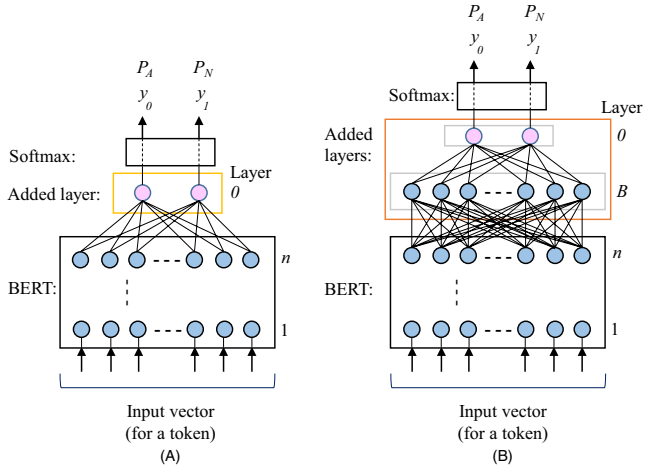

FIGURE 7 Added layers in (A) ZB1 and (B) ZB2 for each input token.

at any position in the document before the location of the ZP predicate, the search space is generally the entire text portion from the beginning of the document to the ZP predicate. The larger is the search space, the more difficult is a ZAR problem.

We introduce the search windows of different sizes. In Figure 8, s_0 denotes a sentence that contains the ZP, and s_0' denotes the part of s_0 from the beginning to the ZP. Symbol s_i denotes the sentence that is the ith sentence to the left of s_0. For a ZP, a search window of size i is considered the portion of the document that comprises the sentences in $\{s_i, s_{i-1}, \ldots, s_1, s_0'\}$. When the window size (WS) is constrained to i, it is assumed that the document starts at s_i, although the document has more sentences in front of s_i. Thus, the search space is limited to a window of size i.

To be fair while comparing two ZAR systems with each other, they should be developed and tested at the same difficulty level. This can be achieved by making both the systems use the same size of search window. This is a widely adopted convention in ZAR research. If we have the performance data of System A, which was developed using the search WS of 3, the same WS should be used to develop and test System B to ensure a fair comparison between both the systems.

The goal of the antecedent-search module discussed in this study is to find a word that is an antecedent of the given ZP. Even if there exist more than one antecedents, finding only one of them is considered to accomplish the goal. Furthermore, in the case of non-anaphoric ZP (ie, there are no antecedents in the search space), the system should recognize that there is no antecedent. However, if an antecedent is found by the system for a non-anaphoric ZP, the system is judged to be wrong in its processing.

Our search module draws inference (prediction) when a ZP is given by using the algorithm depicted in Figure 9. The ZP predicate is placed into Segment A of BERT. The words in the search window are placed in Segment B. The input token sequence is created, as explained in Section 4.2. (In the case of training, the target label sequence is supplied, as explained in Section 4.3.)

In Step 1, BERT will compute $X^{(n)}$, the output of the last layer n. Subsequently, F, the output matrix of layer O at all the positions, will be computed using (10). Finally, Y will be computed from F by using (11). Matrix Y contains probabilities P_A and P_N of labels A and N, respectively, for all the input tokens.

In Step 2, a token that has the highest probability of being classified to label A (ie, representing the maximum likelihood of being an antecedent) is identified. To that end, probabilities P_A (previously mentioned) from the units of output layer O are used. Let $\hat{t}$, $x_{\hat{t}}$, and $\widehat{P}_A$ denote the position, token, and label-A probability of the identified token, respectively.

In Step 3, $\widehat{P}_A$ is compared with threshold γ, where $0 < \gamma \leq 1$, to decide whether token $x_{\hat{t}}$ is from an antecedent. If $\widehat{P}_A \geq \gamma$ and $x_{\hat{t}}$ is a token from a word in Segment B, this word (denoted by wd ($x_{\hat{t}}$)) is considered an antecedent and outputted as the result of the inference algorithm (in line 3-2). If the highest label-A probability, $\widehat{P}_A$, is below threshold γ or the token with $\widehat{P}_A$ is not from a word in Segment B, it is decided that there is no antecedent for the ZP (in line 3-3).

When an index (position) t of a token is given, the corresponding word can be easily found by managing a mapping table between the tokens and words in Segments A and B. If γ is large, the word found by the algorithm might be a correct antecedent. However, a large γ can result in an algorithm that may easily miss finding correct antecedents. Therefore, there exists a trade-off between the value of γ and the quality of algorithm.

To measure the performance, the result of the algorithm for each ZP problem is judged. If the ZP is anaphoric (ie, there are one or more antecedents in the search window), the system should find and output a word that is a correct antecedent specified in the tagged corpus. Otherwise, the system should recognize that there is no antecedent for the ZP.

If the search window is large, Segment B can be big, and thus the input-sequence length can be larger than the maximum allowed input length τ_m of the model. (In our model, it is currently set to 512.) In this case, we split Segment B into several pieces, and each piece is used as Segment B. The model is run for each piece with the same Segment A. The results of these runs are used to obtain a token with the maximum probability $\widehat{P}_A$. This token is used as $x_{\hat{t}}$ in Step 3 of our inference algorithm.

Although the processing of our model is performed at the token level, the inference algorithm should generate an

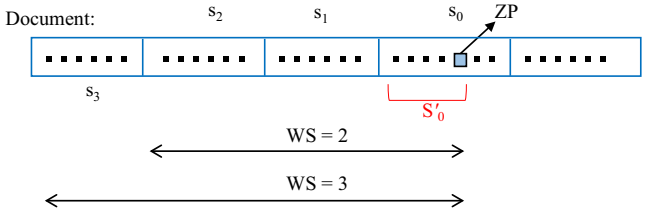

FIGURE 8 Search windows of different sizes.

output at the word level. This results in some complications in system development. Furthermore, the morphemes of words may be chosen as units to be placed into Segments A and B of BERT after performing the morphological analysis of the text. Subsequently, the morphemes will be converted to tokens by using a tokenizer to form an input token sequence.

Figure 10 depicts various schemes of using tokens and morphemes in developing BERT-based ZAR models for Korean. Let word w_i be a correct antecedent of a ZP (which is indicated by an annotation). The target labels assigned to the tokens generated from this word are shown for each scheme.

There are two ways of processing tokens from a word (or morpheme). Let us introduce the FT approach, in which the first token from a word is assigned a special status. It is the main token and represents the word. The other tokens following the first one are considered secondary. Target labels *A* and *N* are assigned to only the first tokens. We introduce a new target label *X*. The secondary tokens are assigned target label *X*. The first and secondary tokens are involved in training by using these target labels. However, for inference, only first tokens are used. In Step 2 of the algorithm, only first tokens are involved in finding a token with maximum P_A. In executing ARGMAX in Step 2-1, the tokens whose target label is *X* are ignored.

The alternative to the FT approach is the AT approach, in which all the tokens of a word (or morpheme) are involved in both training and inference.

When morphemes are fed into Segments A and B, there are the following two options in using the morphemes of words: using only the head morpheme (HM) or using all the morphemes (AM). In the HM scheme, only the HM (ie, the last morpheme of a word in Korean) is used for training and inference. The morphemes that precede an HM in a word are considered secondary. All the tokens from secondary morphemes are assigned label *X* as the target label in training. Although these tokens participate in training, they are ignored in inference like in the FT scheme. Only the tokens from HMs are involved in inference. In the AM scheme, all the morphemes of a word are exploited both in training and inference.

Four possible combinations of token and morpheme usage are depicted in Figure 10B. The HM is denoted by m_2. Notably, token $k_{1,1}$ is the second token of morpheme m_1. In the AT/AM scheme, $k_{1,1}$ participates both in training and inference (its target label is *A*). Thus, if $k_{1,1}$ is identified as $x_{\hat{t}}$ in Steps 2 and 3 and satisfies the test of Step 3, then w_i (which is assigned to variable wd ($x_{\hat{t}}$)) will be the output word. However, in the AT/HM scheme, token $k_{1,1}$ cannot participate in inference because it does not belong to an HM. Despite this, it will be involved in training with target label *X*.

5 | EXPERIMENTATION

We experimentally measured the performance of our models. The results of performance comparison between ours and various other

5.1 | Training data and BERT

We constructed a Korean tagged corpus that can be used for research on Korean ZAR. From Korean Wikipedia, many small text segments were extracted. Each segment comprised approximately 10 consecutive sentences and constituted a document, which is the unit of processing from the ZAR viewpoint. Furthermore, ZPs and their antecedents were tagged in all the documents. The characteristics of our ZAR tagged corpus are listed in Table 1.

Table 2 presents other details regarding our training data. We want to measure system performance using different sizes of search window. The third column shows the maximum number of words in a search window of the corresponding size. The fourth and fifth columns show the number of anaphoric and non-anaphoric ZPs that exist in the entire tagged corpus, respectively. Anaphoricity depends on the given search window. The larger is the search window is, the smaller is the number of non-anaphoric ZPs. Notably, there exist many non-anaphoric ZPs even for a large size of search window. Thus, the approaches in [9] and [10] are problematic because their models aim at resolving only anaphoric ZPs.

We used three versions of BERT for our development and experimentation of our model. The first one is a BERT-base multi-lingual version, which was made publicly available by Google [34]. The other two are the Korean versions of BERT (called KorBERT) and were pre-trained using significant amount of Korean unlabeled texts [35]. One of the two Korean versions uses words as units in Segments A and B. In the other version, morphemes are units in Segment A and B of BERT. Table 3 lists the characteristics of our models that are based on BERT-base. We could not use BERT-large because it requires large memory in GPU. Had we used BERT-large, the performance of our model might have increased by 2% to 3%.

5.2 | Experimental results

We aimed to develop the advanced search module depicted in Figure 2 to be used for ZAR. Because ZP detection is not in the scope of this study, we used gold ZPs in the tagged corpus as input to the search module in our experiments. Therefore, recall and precision are equal to a measure called accuracy, which is computed as follows:

$$\text{acc} = \frac{\#\text{ of ZPs for which systems output is correct}}{\#\text{ of ZPs processed by system}}. \quad (13)$$

For the system to produce a correct output for a given ZP, it should either find one of the (gold) antecedents of the ZP if the ZP is anaphoric, or report "no antecedents" if the ZP is non-anaphoric. Ten-fold cross validation was used in all our experimentations.

The models we experimented are named as follows. BERT versions are distinguished by the first two or three letters of model names. Google BERT is indicated by ZB, which receives only words as input. KorBERT that receives words as input is called KBw, and KorBERT that receives morphemes as input is named KBm. We use letters 1, 2, and L to denote the added architectures of a single layer, two layers, and LSTM-RNN, respectively. Finally, a string that denotes a token/morpheme scheme is appended (see Figure 10). For example, KBmL-AT/AM denotes a model where KorBERT is used; morphemes are input units into segments; LSTM-RNN is added on the top of KorBERT; all the tokens of morphemes are used, and all the morphemes of words are used.

The performance-evaluation results for various models are presented in Table 4. The results can be interpreted as follows.

- Korean BERTs significantly outperformed Google BERT.
- Adding LSTM-RNN improved the performance over feed-forward layers.
- The AT approach is superior to the FT approach.
- Using all the morphemes results in better performance compared with using HMs only.

Input: a ZP and the text in the search window.
1. Model computes matrix Y using Eq. 11 for the input sequence;
2. Find a token whose P_A is the highest among all the input tokens:
2-1. $\hat{t} = \text{ARGMAX}_{t:0\le t\le \tau-1}\ Y_{t,0}$
2-2. $\hat{P}_A = Y_{\hat{t},0}$
2-3. $x_{\hat{t}}$ = token at position $\hat{t}$;
3. If ($\hat{P}_A \ge \gamma$ and $\hat{t}$ belongs to segment B)
then {
3-1. Identify the word $\text{wd}(x_{\hat{t}})$ from which token $x_{\hat{t}}$ is generated;
3-2. Output word $\text{wd}(x_{\hat{t}})$ as an antecedent of the ZP;
}
3-3. else Output message "No antecedents";

FIGURE 9 Inference algorithm (antecedent search).

- The combined scheme of AT/AM achieves higher performance than the other combinations.
- KBmL-AT/AM achieved the best performance with an accuracy of 79.6% for the WS of 3, representing a significant improvement over other works thus far.

The size of layer B affects the model performance, as illustrated in Table 5. The size of 2048 results in the best performance.

Our search algorithm depicted in Figure 9 uses a threshold γ to find an antecedent. We measured the performance of our module while varying γ. The experimental result is depicted in Figure 11. The best performance was achieved when γ was 0.85. This holds for all different sizes of the search window.

To build a ZAR system, we attempted to use two machine-learning models that were not based on BERT. One is an S-SVM, which can be considered one of the best machine-learning models widely used before the era of deep learning [6]. The other is a pointer network, which is a sequence-to-sequence deep-learning model that extensively exploits the attention mechanism [7]. It is one of the advanced deep-learning models. It was successfully used to perform co-reference resolution, which is a problem similar to ZAR in some aspects [8]. The pointer-network-based model used word embeddings created by running open-source utility fastText [36]. An unlabeled text in Korean Wikipedia was supplied to fastText to construct word embeddings. From Table 6, it is evident that our BERT-based models outperformed the S-SVM-based and pointer-network models, respectively.

The drawback of BERT-based models is that they use a significant number of parameters, and thus they can be slow. In Table 6, the inference time (in milliseconds) for resolving a ZP is shown after the slash symbol (/). It is observed that they are more than approximately 15 times slower than the models that do not use BERT.

In Section 2, we described three recent research works on ZAR performed by other researchers [9–11]. These researchers introduced deep-learning models that can be utilized for developing the search module of ZAR. We trained and tested their models using the training data for Korean ZAR. The models were used to build the antecedent-search module in our experiment. The word embeddings used by these three models are the same as the one we used for the above-mentioned pointer-network model. Table 7 presents the performance comparison between these models and ours. This experiment indicates that our models significantly outperform other models.

There exist two approaches of exploiting a pre-trained BERT: fine-tuning and feature based [4]. Our models proposed in this study follow the fine-tuning approach. In the feature-based approach, a pre-trained BERT is used for supplying word embeddings but is not fine-tuned. The performance comparison (in terms of accuracy) between both the

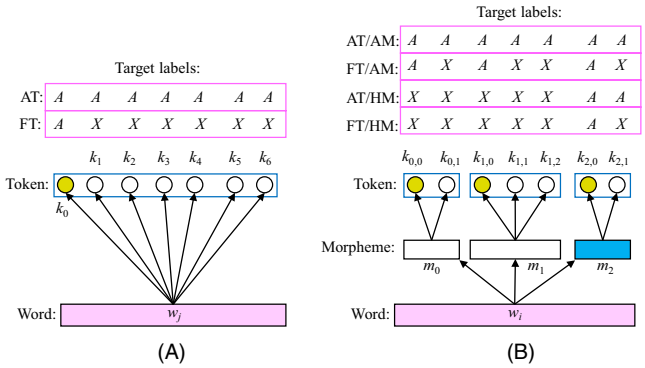

FIGURE 10 Various schemes of exploiting tokens and morphemes. (A) Word units are in segments. (B) Morpheme units are in segments.

TABLE 1 Tagged corpus for Korean ZAR. The second row provides the number of items specified in the first row. (Doc: document)

Doc	Sentence	Word	ZP	ZP/doc	Sentence/doc
532	3135	39 717	2881	5.37	5.88

TABLE 2 Training-data statistics related with the size of search window.

Window size	ZPs	Max length	Anaphoric	Non-anaphoric
0	2870	73	1034	1836
1	2870	93	1551	1319
2	2870	109	1801	1069
3	2870	124	1937	933

approaches is presented in Table 8. Evidently, the fine-tuning approach outperforms the feature-based one.

6 | DISCUSSIONS

Our model is appropriate for sequence labeling. The input is a sequence of words in the text. The output is a label sequence, where each output label is assigned to each token of the input sequence as depicted in Figure 12A. An output label at a position in the sequence is determined by considering the entire context of the text and labels at other positions, meaning that all the tokens are simultaneously considered.

However, the models proposed in other works follow a candidate-wise approach [9–11]. They process one CA at a time separately, as depicted in Figure 12B. A label assigned to a CA does not affect the labels of otherCAs. Notably, the pointer network outperforms the systems in other works. The pointer network is a sequence model that comprises encoding and decoding sequences.

TABLE 3 Specifications of our BERT-based model.

Pre-training	Hidden layer size	768
	Max. sequence length	512
	Num. of layers	12
	Num. of attention heads	12
	Vocab. size (Google BERT)	105 879
	Vocab. size (Korean BERT)	30 371
	Num. of parameters	110 M
Fine-tuning	Size of layer B	2 018
	Size of layer O	2
	Size of the hidden layer of LSTM	4096
	Max. sequence length τ_m	512

To ensure fairness while comparing the competing models and follow the recent trend of deep learning, the end-to-end learning paradigm is enforced in the training and testing of all the models in our experimentations. The input to the models is no more than a word sequence that appears in the text. Handcrafted features were used in the past models introduced in [9,10]. However, such features are not used to implement their models in our experiment (see Figure 7). Syntactic-analysis information was exploited in [11]. For a similar reason, such information is not used in our implementation of their model in [11] to conduct a performance comparison (see Table 7).

According to our experiments, our model performs better than other models in Korean ZAR. This might be attributed to the following.

- BERT is a general language representation model, which powerfully models the deep bidirectional contexts of a language.
- BERT is pre-trained using a large unlabeled corpus, and thus it can capture the predicate-argument semantics of a language.
- The deep-learning attention mechanism used by BERT (and Transformer) can model long-distance dependencies in the text.
- The sequence-labeling approach applied to ZAR is effective in utilizing contextual information.

To qualitatively evaluate our model, qualitative assessment, such as indicating strengths and weaknesses, must be performed. Our model can achieve high performance, despite the fact that it is fine-tuned using small training data (comprises only 2881 ZPs). The performance can be further enhanced by increasing the size of the tagged corpus used for fine-tuning. Our model requires minimal feature-engineering for training and inference. Because the pre-training of BERT is unsupervised, the performance might be improved further by increasing the size of unlabeled data. However, one

TABLE 4 Performance in terms of the accuracy of our models (γ = 0.85).

WS	0	1	2	3
ZB1-AT	80.6	70.4	66.9	66.1
ZB2-AT	81.0	72.0	68.5	66.9
ZBL-FT	78.2	67.5	61.9	61.1
ZBL-AT	81.2	75.3	73.9	73.0
KBw2-AT	86.1	77.4	77.1	75.3
KBwL-FT	84.7	76.7	76.4	76.0
KBwL-AT	86.1	79.6	78.8	77.9
KBmL-AT/HM	84.6	78.7	76.5	76.4
KBmL-AT/AM	85.8	80.5	79.8	79.6

TABLE 5 Effect of the size of layer *B* on our model ZB2-AT.

Size of layer *B*	32	128	1024	2048	3072
WS = 0	78.0	78.8	79.6	81.0	81.1
WS = 1	68.2	68.8	70.6	72.0	70.6
WS = 2	61.9	62.4	67.1	68.5	68.0
WS = 3	59.5	60.2	65.5	66.9	66.9

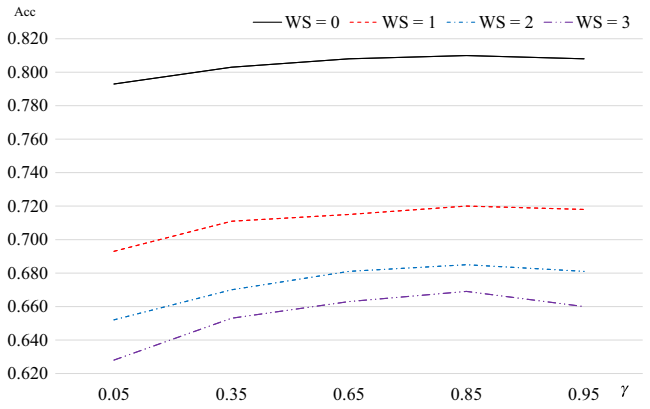

FIGURE 11 Performance curves obtained by varying threshold γ (WS: size of the search window). The model used is ZB2-AT.

of the weaknesses of our approach is that constructing significantly large tagged data requires considerable time and efforts. Second, useful heuristic information, which may be effective for improving the performance, is not exploited in our approach. Third, extensive computation is required for training and inference, and thus powerful hardware resources are required to develop and run the model.

Let us use two ZP problems depicted in Figure 13 as a case study to illustrate the working of our model. In Figure 13A, a ZP occurs at word 시작하였다 because its subject is missing. All the words in the document are fed to Segment B because WS = 3. The models of past researches only considered NPs and usedhem as CAs. However, our model does

TABLE 6 Comparison between models with BERT and those without BERT. Each cell represents accuracy/inference time (ms).

WS	0	1	2	3
S-SVM	62.4/2	52.5/3	45.2/4	40.1/5
Pointer Net.	73.7/2	63.7/3	63.4/4	63.7/5
ZBL-AT	81.2/60	75.3/64	73.9/70	73.0/76
KBmL-AT/AM	85.8/63	80.5/70	79.8/77	79.6/82

TABLE 7 Comparison among research works in terms of accuracy.

WS	0	1	2	3
Chen & Ng [9]	59.7	45.6	37.9	33.6
Yin et al. [10]	70.1	52.7	44.8	40.8
Iida et al. [11]	73.3	62.7	58.3	56.6
ZBL-AT	81.2	75.3	73.9	73.0
KBmL-AT/AM	85.8	80.5	79.8	79.6

not follow this approach. All the words in the search window are considered in our model. For example, word 지낸 is processed even if it is a verb.

The first word 자르다리는 is tokenized to tokens 자, 르, 다리, and 는_, as shown in the parentheses. The probability P_A of antecedent label of each token is computed by the model. The maximum P_A of all the tokens from each word is shown just above the word in Figure 13.

The P_A of token 다리 is maximum among the P_A s of all the tokens in the input sequence. It is greater than threshold γ. Because 다리 belongs to word 자르다리는, 자르다리는 is determined to be an antecedent. In this example, our model assigns significantly low P_A to the tokens of words that are not antecedents. It assigns considerably high P_A only to the tokens of a word that is an antecedent. This means that our model works correctly. Although there exist satisfactory candidates as antecedent, such as 부토가 and 대통령이, our model does not assign high P_A to the tokens from them. This could be possible because BERT can utilize long-distance contexts.

Another ZP example is depicted in Figure 13B. A ZP occurs at predicate 도착했다. Because there is no token whose P_A is greater than threshold $\gamma = 0.85$, the inference algorithm determines that there is no antecedent for the given ZP. This decision is correct. Our model avoided making a mistake of selecting 갈릴레이의 as an antecedent. Furthermore, "갈릴레이가 도착했다" is a fine sentence in Korean. However, in the context of this document, 갈릴레이 is not appropriate to be a subject of 도착했다. There is no antecedent for the ZP in this document, and thus it is non-anaphoric. Our model is intelligent to make a correct decision for this hard problem, and this is surprising.

The end-to-end learning paradigm is adopted by our model, as previously mentioned. Raw text is supplied as

TABLE 8 Comparison of fine-tuning and feature-based approaches (WS = 3).

	Feature-based	Fine-tuning
ZBL-AT	71.2	73.0
KBwL-AT	77.8	77.9
KBmL-AT/AM	77.4	79.6

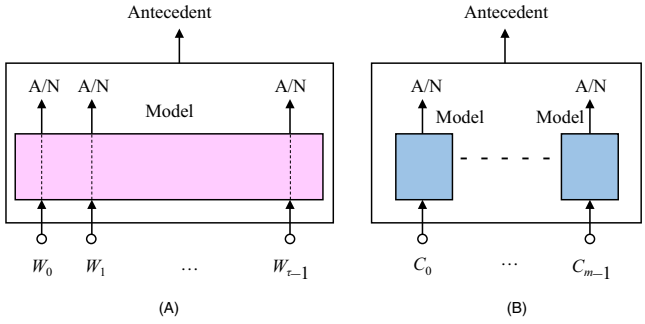

FIGURE 12 Model types for ZAR. w_i: an input token; C_i: a CA.

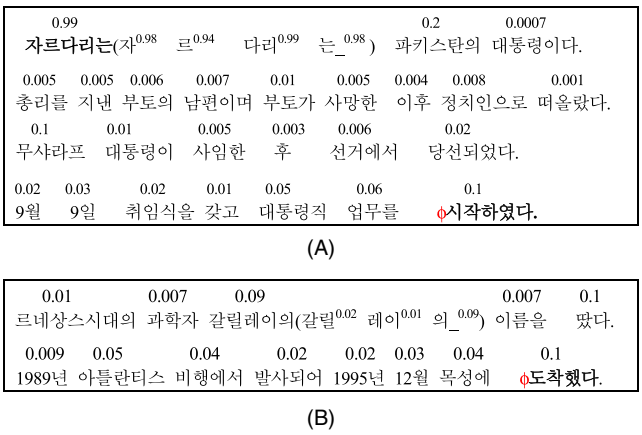

FIGURE 13 Case study of running our model. To save space, words are displayed even if the processing occurs at the token level; we show the tokens for the following two words: 자르다리는 and 갈릴레이의

input to our algorithm. Subsequently, the algorithm produces a word as an answer. Our algorithm does not exploit information, such as parts of speech and parse trees. All the words (or morphemes) in the text of search window are treated in the same way without considering information, such as parts of speech and phrases. In other works, only NPs were chosen and considered as CAs during the processing. However, all words are considered in our approach.

7 | CONCLUSION

We presented a deep-learning model that could improve ZAR. Our main idea was to utilize a general language representation model called BERT. We chose fine-tuning as our approach of exploiting BERT. Beginning from a pre-trained BERT, which was publicly available, our model was fine-tuned using training data prepared from a Korean tagged corpus for ZAR.

We built various machine-learning models designed by us and others to make performance comparisons. The experimental results showed that our model significantly improved the performance over other models. Moreover, our model complied with the end-to-end learning paradigm, which was not pursued by other works.

We plan to do research on employing deep-learning models for ZP detection in the future. Adding more advanced architectures to BERT will also be pursued to improve the ZAR performance of our BERT-based models. Building our ZAR models based on new advanced models, such as RoBERTa, ALBERT, and XLNet, is included in our future research plan [20–22].

ORCID

Dongyul Ra https://orcid.org/0000-0003-1449-4614

REFERENCES

1. R. Sasano, D. Kawahara, and S. Kurohashi, *A fully-lexicalized probabilistic model for Japanese zero anaphora resolution*, in Proc. 22nd Int. Conf. Comput. Linguistics (Manchester, UK), Aug. 2008, pp. 769–776.
2. H. Nakaiwa and S. Shirai, *Anaphora resolution of Japanese zero pronouns with deictic reference*, in Proc. Int. Conf. Comput. Linguistics. (Madrid, Spain), 1996, pp. 812–817.
3. R. Iida, I. Kentaro, and Y. Matsumoto, *Zero anaphora resolution by learning rich syntactic pattern features*, ACM Trans. Asian Language Inform. Process. **6** (2007), no. 4, 12:1–22.
4. J. Devlin et al., *BERT: Pre-training of deep bidirectional transformers for language understanding*, in Proc. North American Chap. Assoc. Comput. Linguistics (Minneapolis, MN, USA), 2019, pp. 4171–4186.
5. A. Vaswani et al., *Attention is all you need*, Advances Neural Inform. Process. Syst. **30** (2017), 6000–6010.
6. I. Tsochantaridis et al., *Support vector machine learning for interdependent and structured output spaces*, in Proc. 21st Int. Conf. Mach. Learn. (Banff, Canada), 2004, pp. 823–830.
7. O. Vinyals, M. Fortunato, and N. Jaitly, *Pointer networks*, Advances in Neural Inform. Process. Syst. **28** (2015), 2674–2682.
8. C. Park and C. Lee, *Co-reference resolution for Korean pronouns using pointer networks*, J. Korean Inst. Inform. Sci. Eng. **44** (2017), 496–502.
9. C. Chen and V. Ng, *Chinese zero pronoun resolution with deep neural networks*, in Proc. Annu. Meet. Assoc. Comput. Linguistics, (Berlin, Germany), 2016, pp. 778–788.
10. Q. Yin et al., *Zero pronoun resolution with attention-based neural network*, in Proc. Int. Conf. Comput. Linguistics (Santa Fe, NM, USA), 2018, pp. 13–23.
11. R. Iida et al., *Intra-Sentential subject zero anaphora resolution using multi-column convolutional neural network*, in Proc. Conf. Empirical Methods in Natural Language Process. (Austin, TX), 2016, pp. 1244–1254.
12. M. Okumura and K. Tamura, *Zero pronoun resolution in Japanese discourse based on centering theory*, in Proc. Int. Conf. Computat. Linguistics. (Copenhagen, Denmark), 1996, pp. 871–887.

13. M. Murata and M. Nagao, *Resolution of verb ellipsis in Japanese sentences using surface expressions and examples*, in Proc. Natural Language Process. Pacific Rim Symp. (Bangkok, Thailand), 1997, pp. 75–80.
14. S. Nariyama, *Grammar for ellipsis resolution in Japanese*, in Proc. Int. Conf. Theoret. Methodol. Issues Mach. Transl. (Keihanna, Japan), 2020, pp. 135–145.
15. K. Seki, A. Fujii, and T. Ishikawa, *A probabilistic method for analyzing Japanese anaphora integrating zero pronoun detection and resolution*, in Proc. Int. Conf. Comput. Linguistics (Taipei, Taiwan), 2002, pp. 911–917.
16. S. Lim, C. Lee, and M. Jang, *Restoring an elided entry word in a sentence for encyclopedia QA system*, in Proc. Int. Joint Conf. Natural Language Process. (Jeju, Rep. of Korea), 2005, pp. 215–219.
17. S. Zhao and H. T. Ng, *Identification and resolution of Chinese zero pronouns: A machine learning approach*, in Proc. Joint Conf. Empirical Methods Natural Language Process. Comput. Natural Language Learn. (Prague, Czech), 2007, pp. 541–550.
18. R. Iida and M. Poesio, *A cross-lingual ILP solution to zero anaphora resolution*, in Proc. Annu. Meet. Assoc. Comput. Linguistics (Portland, OR, USA), 2011, pp. 804–813.
19. S. Jung and C. Lee, *Deep neural architecture for recovering dropped pronouns in Korean*, ETRI J. **40** (2018), 257–264.
20. Z. Lan et al., *ALBERT: A lite BERT for self-supervised learning of language representations*, in Proc. Int. Conf. Learn. Represent. (Addis Ababa, Ethiopia), 2020.
21. Z. Yang et al., *XLNet: Generalized autoregressive pretraining for language understanding*, arXiv preprint, CoRR, 2020, arXiv:1906.08237.
22. Y. Liu et al., *Roberta: A robustly optimized BERT pretraining approach*, arXiv preprint, CoRR, 2019, arXiv:1907.11692.
23. Q. Yin et al., *Chinese zero pronoun resolution: A collaborative filtering-based Approach*, ACM Trans. Asian Low-Resour. Lang. Inf. Process. **19** (2019) no. 1, 3:1–20.
24. F. Kong, M. Zhang, and G. Zhou, *Chinese zero pronoun resolution: A chain-to-chain approach*, ACM Trans. Asian Low-Resour. Lang. Inf. Process. **19** (2019), no. 1, 2:1–21.
25. I. Goodfellow, Y. Bengio, and A. Courville, Deep learning, MIT Press, Cambridge, MA, USA 2016.
26. S. Hochreiter, and J. Schmidhuber, *Long short-term memory*, Neural Comput. **9** (1997), 1735–1780.
27. J. Chung et al., *Empirical evaluation of gated recurrent neural networks on sequence modeling*, in Proc. Neural Inform. Process. Syst., Workshop Deep Learn. Dec. 2014.
28. I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to sequence learning with neural networks*, in Proc. Int. Conf. Neural Inform. Process. Syst. (Montreal, Canada), 2014, pp. 3104–3112.
29. D. Bahdanau, K. H. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, in Proc. Int. Conf. Learn. Represent. (San Diego, CA, USA), 2015.
30. J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, ArXiv Preprint, CoRR, 2016, ArXiv:1607.06450.
31. Y. Wu et al., *Google's neural machine translation system: Bridging the gap between human and machine translation*, arXiv preprint, 2016, arXiv:1609.08144.
32. M. Schuster and K. Nakajima, *Japanese and Korean voice search*, in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (Kyoto, Japan), 2012, pp. 5149–5152.
33. R. Sennrich, B. Haddow, and A. Birch, *Neural machine translation of rare words with subword units*, in Proc. Annu. Meet. Assoc. Comput. Linguistics (Berlin, Germany), 2016, pp. 1715–1725.
34. Google, *TensorFlow code and pre-trained models for BERT*, available at https://github.com/google-research/bert.
35. ETRI, *Public Artificial Intelligence Open API DATA*, *Korean BERT language model*, available at http://aiopen.etri.re.kr/service_dataset.php. (Korean).
36. Facebook, *fastText-Library for text representation and classification*, available at https://fasttext.cc/.

AUTHOR BIOGRAPHIES

Youngtae Kim received his BS degree in computer science from Halla University, Wonju, Rep. of Korea, in 2008. He worked for Tomato System as a technical staff until 2009. He is pursuing his PhD degree at Yonsei University, Wonju, Rep. of Korea. His main research interests are natural-language processing, artificial intelligence, information retrieval, and question answering.

Dongyul Ra received his BS degree in electronics engineering from Seoul National University, Seoul, Rep. of Korea, in 1978, MS degree in computer science from KAIST, Daejeon, Rep. of Korea, in 1980, and PhD degree in computer science from Michigan State University, USA, in 1989. Since 1991, he has been a faculty member with Yonsei University, Wonju, Rep. of Korea. His main research interests are natural-language processing, artificial intelligence, and data mining.

Soojong Lim received his BS degree in mathematics from Yonsei University, Seoul, Rep. of Korea, in 1997, and his MS and PhD degrees in computer science from Yonsei University, in 1998 and 2014, respectively. He is currently a principal researcher at ETRI, Daejeon, Rep. of Korea. His main research interests are natural-language processing, question answering, machine learning, and deep learning.