Received: 7 December 2020 | Revised: 8 July 2021 | Accepted: 23 August 2021

DOI: 10.4218/etrij.2020-0460

ORIGINAL ARTICLE

ETRI Journal WILEY

DG-based SPO tuple recognition using self-attention M-Bi-LSTM

Joon-young Jung

Artificial Intelligence Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

Correspondence
Joon-young Jung, Artificial Intelligence Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.
Email: jyjung21@etri.re.kr

Funding information
Electronics and Telecommunications Research Institute, Grant/Award Number: 21ZS1100

Abstract

This study proposes a dependency grammar-based self-attention multilayered bidirectional long short-term memory (DG-M-Bi-LSTM) model for subject–predicate–object (SPO) tuple recognition from natural language (NL) sentences. To add recent knowledge to the knowledge base autonomously, it is essential to extract knowledge from numerous NL data. Therefore, this study proposes a high-accuracy SPO tuple recognition model that requires a small amount of learning data to extract knowledge from NL sentences. The accuracy of SPO tuple recognition using DG-M-Bi-LSTM is compared with that using NL-based self-attention multilayered bidirectional LSTM, DG-based bidirectional encoder representations from transformers (BERT), and NL-based BERT to evaluate its effectiveness. The DG-M-Bi-LSTM model achieves the best results in terms of recognition accuracy for extracting SPO tuples from NL sentences even if it has fewer deep neural network (DNN) parameters than BERT. In particular, its accuracy is better than that of BERT when the learning data are limited. Additionally, its pretrained DNN parameters can be applied to other domains because it learns the structural relations in NL sentences.

KEYWORDS
dependency grammar, information extraction, long short-term memory, SPO tuple

1 | INTRODUCTION

There is a large amount of information on the Internet, and large-scale knowledge bases (KBs), such as DBpedia [1], Wikidata [2], and Yago [3], have been built using this information. These KBs are used in artificial intelligence research, including natural language processing (NLP). However, it is not easy to expand knowledge because the cost and effort required to expand new knowledge manually are high. Because these KBs need to be expanded to include new knowledge, despite the difficulty, research on KB expansion at low cost has been conducted [4–6].

To add new knowledge to KBs at a low cost, KBs should autonomously expand its knowledge. It is necessary to extract knowledge from newly created data and expand a KB with the extracted knowledge for autonomous knowledge growth. Therefore, it is essential to extract knowledge from several data for autonomous knowledge growth.

To extract knowledge from several data, it is necessary to extract knowledge from structured and unstructured data. Because knowledge can effectively be extracted from structured data, research has been conducted to extract knowledge from such data [7–9].

However, a large amount of new and diverse knowledge exists in the form of unstructured data. Although it is difficult to extract knowledge from unstructured data, research on extracting knowledge from unstructured data has been conducted [10–29]. Recent deep neural network (DNN) models that improve the performance of knowledge extraction from unstructured data require many training data and have many DNN parameters to be trained. Therefore, it takes a long time to train a high-accuracy DNN model, such as bidirectional encoder representations from transformers (BERT) proposed by Devlin et al. [30]. To extract knowledge from unstructured data using the DNN model with a small amount of training data and a small number of DNN parameters, this study proposes a dependency grammar (DG)-based self-attention multilayered bidirectional long short-term memory (DG-M-Bi-LSTM) model. The proposed model does not require complex DNN models and large learning data because it uses nonvariable DG relations as inputs. The parameter size of the proposed model is only 15 M, whereas the parameter size of the BERT model is 340 M. Additionally, the top-5 accuracies of the proposed model and BERT are 0.948 and 0.705, respectively, using limited learning data (10 000). The proposed method can apply the pretrained DNN parameters to other domains by learning the structural relations in sentences instead of learning the semantic relations of specific words.

The remainder of the paper is organized as follows. Section 2 describes related work. Section 3 describes the DG-M-Bi-LSTM model for end-to-end subject–predicate–object (SPO) tuple recognition. In Section 4, we present the results of our experiments. Finally, Section 5 presents the conclusion.

2 | RELATED WORK

Research on information extraction (IE) from natural language (NL) sentences has been extensively conducted in NLP. To increase usability in various domains, a method for extracting a relational tuple from an open domain instead of a specific domain is being studied. The open IE methods are as follows. Mintz et al. [10] proposed extracting tuples based on distance supervision using an unlabeled corpus. However, the distance supervision method may yield low accuracy owing to noisy patterns. Riedel et al. [11] reduced the relation extraction error using a factor graph and constraint-driven semisupervision. They generated latent feature vectors of entity tuples and relations using a matrix factorization model [12]. To improve the wrong label problem of distant supervision-based relation extraction, Zeng et al. [13] proposed piecewise convolutional neural networks; Lin et al. [14] proposed a sentence-level attention-based model; and Zhou et al. [15] proposed a hierarchical selective attention network to reduce the high computational cost of distant supervision-based relation extraction.

Banko et al. [16] proposed TextRunner to extract many relational tuples from the corpus through a three-step pipeline structure. TextRunner is limited in that it extracts incorrect tuples due to incoherent and uninformative extractions. Fader et al. [17] proposed a syntactic and lexical constraint added method (ReVerb) to extract correct tuples. However, ReVerb is limited in that it extracts only relations according to verbs and extracts tuples that are not true because it ignores context. Schmitz et al. [18] extracted relations according to nouns or adjectives and added a context-analysis step to overcome ReVerb's limitation.

Corro et al. [19] proposed ClausIE to analyze clauses in a sentence and extract relations and arguments from NL sentences using dependency paths. Gashteovski et al. [20] improved overly specific extractions in ClausIE and provided semantic annotations for each extraction to increase precision and recall performance. Angeli et al. [21] proposed extracting clauses from a long sentence using a clause splitter with distant training and extracting a triple by running natural logic inference over these extracted clauses.

Recently, relation extraction using DNNs has been studied. Stanovsky et al. [22] extracted open IE tuples by supervised learning using a bidirectional long short-term model (Bi-LSTM) and semantic role labeling models. Cui et al. [23] proposed a neural open IE using an encoder–decoder framework. Sun et al. [24] proposed an end-to-end neural model using the sequence-to-sequence paradigm to convert NL sentences into facts. Jia et al. [25] proposed a hybrid neural sequence-tagging model using Bi-LSTM, a convolutional neural network, and a conditional random field for relation extraction. Jiang et al. [26] proposed an iterative rank-aware learning method to increase the precision and recall of extraction tuples. Trisedya et al. [27] proposed an end-to-end relation extraction model for KB enrichment using a neural encoder–decoder model. Shi et al. [28] used BERT and LSTM models to extract relations and label semantic roles. Papanikolaou et al. [29] proposed a method to extract relations in unsupervised scenarios using a pretrained BERT model.

In recent studies, complex DNN models, such as BERT, are trained using a large amount of training data to improve the accuracy of tuple extraction. However, this study proposes DG-M-Bi-LSTM, which extracts SPO tuples from NL sentences with high accuracy even though the training dataset is small, and the DNN model is not complex.

3 | DG-BASED SELF-ATTENTION M-BI-LSTM

The DNN model for recognizing SPO tuples consists of DG embedding, multilayered bidirectional LSTM (M-Bi-LSTM), self-attention, and SPO tuple classification (Figure 1).

The DG embedding parses the DG for NL and embeds the generated DG relations. M-Bi-LSTM receives the DG embedding as input vectors and performs deep learning using multilayered bidirectional LSTM. Self-attention calculates the attention scores for each step of M-Bi-LSTM and generates attention values. The SPO tuple classification determines the SPO tuples of the NL sentences. The details of each module are as follows.

3.1 | DG embedding

To extract knowledge from NL sentences, it is necessary to understand NL sentences through syntactic and semantic analysis. The syntactic analysis of NL sentences can be used to extract knowledge regardless of the semantic meaning of the NL sentences. The pretrained DNN parameters can be used in different domains for knowledge extraction if the DNN model is trained using syntactic analysis data. Therefore, research is underway to understand the syntactical structure of NL sentences [31–34]. The DG is a syntactic parsing that returns a parse tree for a sentence. A parse tree has directed syntactic relationships between words in a sentence. It means that directed links represent the lexical dependencies between words in a sentence. Chen et al. [35] proposed a fast and accurate dependency parser using a greedy, transition-based method. This dependency parser is used to discover the syntactical structures of NL sentences. Figure 2 shows the DG relations in the NL sentence (Google has announced their Android platform for mobile devices). Universal dependencies provide an inventory of DG relations and contain 40 DG relations between words [32].

To learn knowledge extraction based on DG, the learning data in the form of DG should be generated. Then, the DNN model for NLP should be trained using the generated DG relations. Therefore, the generated DG relations using a dependency parser should be embedded. Several methods have been proposed for word embedding. Cbow [36] and Skip-gram [37] used n-grams, and GloVe [38] used the probability of co-occurrence of words. Skip-gram is used for DG embedding because DG relations are not diverse, and Skip-gram works well even though it is not complicated.

3.2 | M-Bi-LSTM

The recurrent neural network (RNN) was proposed for sequential data [39]. In RNN, the input is a sequence of vectors, and the output represents some information about the sequence at every step in the input. Because RNN maintains a vector of activation for each timestep, it is used for NLP applications [40,41]. However, the output sequence of RNN is biased toward the most recent inputs in the sequence [42]. To overcome this vanishing gradient problem in RNNs, the LSTM network is used to obtain long-range dependencies. The initial version of the LSTM consists of memory cells, input gate, and output gate. However, it excludes the forget gate and peephole connections [43]. Gers et al. [44] proposed an LSTM architecture that includes the forget gate, enabling LSTM to reset its state. They introduced peephole connections to make precise timings easier to learn [45]. Graves et al. [46] presented the most commonly used LSTM architecture using full backpropagation-through-time training for an LSTM network.

The LSTM is adopted as a part of the proposed network structure because long-term dependencies could exist between words in a long NL sentence. To increase the performance of LSTM, Bi-LSTM concatenates forward and backward LSTM-hidden states (HS). Bi-LSTM is made by combining bidirectional RNN [47] with

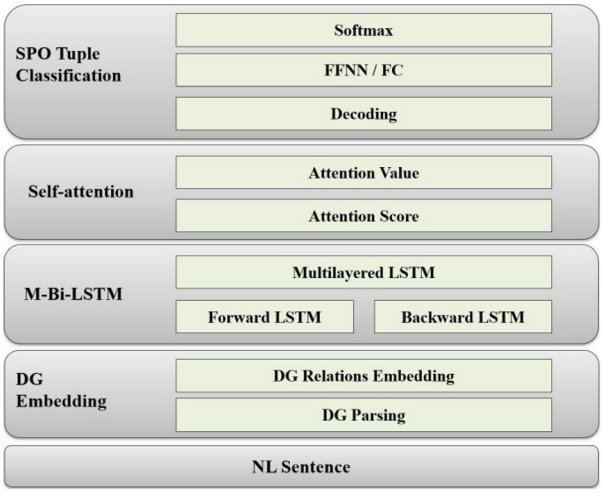

FIGURE 1 DG-M-Bi-LSTM model

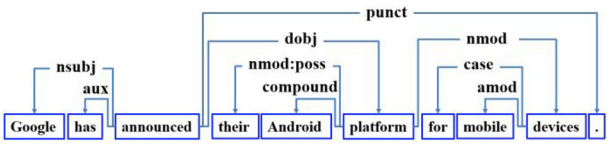

FIGURE 2 Example of DG relations in an NL sentence

LSTM. Bi-LSTM processes input vector sequences in the forward and backward directions and performs substantially better than unidirectional LSTM [48,49]. The forward hidden sequence is calculated by iterating the forward layer from $t = 1$ to T as follows (1):

$$\overrightarrow{\boldsymbol{h}}_t = H\left(\boldsymbol{W}_{x\overrightarrow{h}}\boldsymbol{x}_t + \boldsymbol{W}_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_{\overrightarrow{h}}\right). \quad (1)$$

However, the backward hidden sequence is calculated by iterating the backward layer from $t = T$ to 1 as follows (2):

$$\overleftarrow{\boldsymbol{h}}_t = H\left(\boldsymbol{W}_{x\overleftarrow{h}}\boldsymbol{x}_t + \boldsymbol{W}_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{\boldsymbol{h}}_{t+1} + \boldsymbol{b}_{\overleftarrow{h}}\right). \quad (2)$$

The forward and backward LSTM-hidden layers are combined as follows (3):

$$\boldsymbol{h}_t = \Phi\left(\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t\right). \quad (3)$$

Here, $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ are the forward hidden state (FHS) and backward hidden state (BHS) at time t, respectively; $\boldsymbol{W}_{x\overrightarrow{h}}$ and $\boldsymbol{W}_{x\overleftarrow{h}}$ are the weight matrices mapping the input vector to FHS and BHS, respectively. Additionally, $\boldsymbol{W}_{\overrightarrow{h}\overrightarrow{h}}$ and $\boldsymbol{W}_{\overleftarrow{h}\overleftarrow{h}}$ are the weight matrices mapping the previous FHS to FHS and the previous BHS to BHS, respectively; $\boldsymbol{b}_{\overrightarrow{h}}$ and $\boldsymbol{b}_{\overleftarrow{h}}$ are the bias vectors of FHS and BHS, respectively; H is the hidden layer function; and Φ is the function that combines the states.

Multiple hidden layers can also be stacked to increase the performance of LSTM. Multilayered LSTM (M-LSTM) stacks multiple hidden layers and sends HS sequences from the layer below the upper layer as follows (4):

$$\boldsymbol{h}_t^n = H\left(\boldsymbol{W}_{h^{n-1}h^n}\boldsymbol{h}_t^{n-1} + \boldsymbol{W}_{h^nh^n}\boldsymbol{h}_{t-1}^n + \boldsymbol{b}_h^n\right). \quad (4)$$

Here, $\boldsymbol{h}_t^n$, $\boldsymbol{h}_t^{n-1}$, and $\boldsymbol{h}_{t-1}^n$ are the HS at time t on the n^{th} layer, HS at time t on the lower layer, and HS at time $t-1$ on the n^{th} layer, respectively. $\boldsymbol{W}_{h^{n-1}h^n}$ and $\boldsymbol{W}_{h^nh^n}$ are the weight matrices mapping HS on the lower layer to HS on the n^{th} layer and mapping the previous HS to HS on the n^{th} layer, respectively. Note that $\boldsymbol{h}_t^0 = \boldsymbol{x}_t$ and $\boldsymbol{b}_h^n$ is the bias vector of HS on the n^{th} layer.

Furthermore, Bi-LSTM and M-LSTM are combined to increase the performance of Bi-LSTM or M-LSTM alone. M-Bi-LSTM can be implemented by stacking FHS and BHS, as shown in (5) to (7).

$$\overrightarrow{\boldsymbol{h}}_t^n = H\left(\boldsymbol{W}_{\overrightarrow{h}^{n-1}\overrightarrow{h}^n}\overrightarrow{\boldsymbol{h}}_t^{n-1} + \boldsymbol{W}_{\overrightarrow{h}^n\overrightarrow{h}^n}\overrightarrow{\boldsymbol{h}}_{t-1}^n + \boldsymbol{b}_{\overrightarrow{h}}^n\right), \quad (5)$$

$$\overleftarrow{\boldsymbol{h}}_t^n = H\left(\boldsymbol{W}_{\overleftarrow{h}^{n-1}\overleftarrow{h}^n}\overleftarrow{\boldsymbol{h}}_t^{n-1} + \boldsymbol{W}_{\overleftarrow{h}^n\overleftarrow{h}^n}\overleftarrow{\boldsymbol{h}}_{t+1}^n + \boldsymbol{b}_{\overleftarrow{h}}^n\right), \quad (6)$$

$$\boldsymbol{h}_t^N = \Phi\left(\overrightarrow{\boldsymbol{h}}_t^N, \overleftarrow{\boldsymbol{h}}_t^N\right). \quad (7)$$

Here, $\overrightarrow{\boldsymbol{h}}_t^n$ and $\overleftarrow{\boldsymbol{h}}_t^n$ are the FHS and BHS at time t on the n^{th} layer, respectively; $\boldsymbol{W}_{\overrightarrow{h}^{n-1}\overrightarrow{h}^n}$ is the weight matrix mapping FHS on the lower layer to the FHS on the n^{th} layer; $\boldsymbol{W}_{\overleftarrow{h}^{n-1}\overleftarrow{h}^n}$ is the weight matrix mapping BHS on the lower layer to BHS on the n^{th} layer; $\boldsymbol{W}_{\overrightarrow{h}^n\overrightarrow{h}^n}$ is the weight matrix mapping the previous FHS to FHS on the n^{th} layer; $\boldsymbol{W}_{\overleftarrow{h}^n\overleftarrow{h}^n}$ is the weight matrix mapping the previous BHS to BHS on the n^{th} layer. Furthermore, $\boldsymbol{b}_{\overrightarrow{h}}^n$ and $\boldsymbol{b}_{\overleftarrow{h}}^n$ are the bias vectors of FHS and BHS on the n^{th} layer, respectively.

Figure 3 shows the example of M-Bi-LSTM. The input of M-Bi-LSTM is a sequence of DG embeddings.

3.3 | Self-attention

Attention is performed to highlight the part of the sentence related to the SPO tuple relation recognition. The attention score of each step is obtained by correlating the output of the last step of M-Bi-LSTM with the output of each step, and attention is performed by applying the attention score to the output of each step. When different input feature vectors are processed, the attention mechanism can focus on different parts of a sentence to optimize the process of the deep learning task. Attention mechanisms obtain superior results in image recognition [50], machine translation [51], sentence summarization [52], and text classification

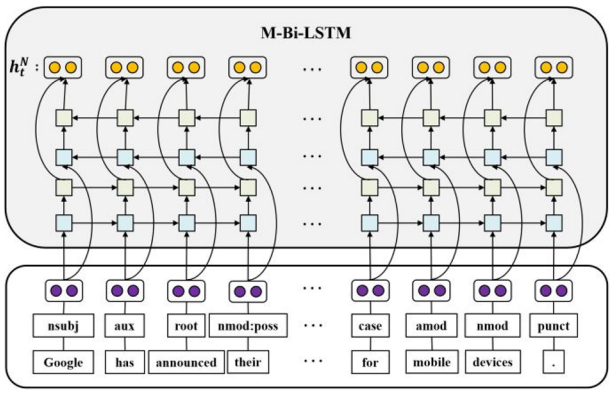

FIGURE 3 M-Bi-LSTM diagram

[53–55]. Self-attention is an attention mechanism relating different positions of a single sequence to compute a representation of the sequence [56]. Self-attention performs well in several tasks, such as machine reading [57] and summarization [58].

Therefore, this study proposes a self-attention mechanism to attend to the important part of a sentence (Figure 4). As a result, the SPO tuple classification for knowledge extraction can concentrate on the key parts of a sentence. Here m_t is the attention mask, a_t is the attention score, and $\boldsymbol{v}_t$ is the attention value for the SPO tuple classification, as expressed in (8) to (10), respectively:

$$m_t = \begin{cases} 0\,(t \le \text{len(sentence)}) \\ -\text{Inf}\,(t > \text{len(sentence)}) \end{cases}, \tag{8}$$

$$a_t = \tanh\left\{\boldsymbol{W}_a\left(\boldsymbol{h}_t^N \bullet \boldsymbol{h}_{\overline{T}}^N\right) + \boldsymbol{b}_a\right\} \odot m_t, \tag{9}$$

$$\boldsymbol{v}_t = \boldsymbol{h}_t^N \bullet \frac{\exp(a_t)}{\sum_{j=1}^{T}\exp(a_j)}. \tag{10}$$

Here, $\boldsymbol{h}_t^N$ is the HS at time t on the last layer; $\boldsymbol{h}_{\overline{T}}^N$ is the HS at the last word of the sentence on the last layer. The symbol $\bullet$ denotes matrix multiplication, and $\odot$ is the element-wise addition. Furthermore, $\overline{T}$ is the sequence time of the last word of the sentence; $\boldsymbol{W}_a$ is the weight matrix mapping the matrix multiplication of $\boldsymbol{h}_t^N$ and $\boldsymbol{h}_{\overline{T}}^N$ to the attention score; and $\boldsymbol{b}_a$ is the bias of the attention score.

3.4 | SPO tuple classification

The M-Bi-LSTM and self-attention values are used for the SPO tuple classification (Figure 5). First, self-decoding is performed by applying the result of the last

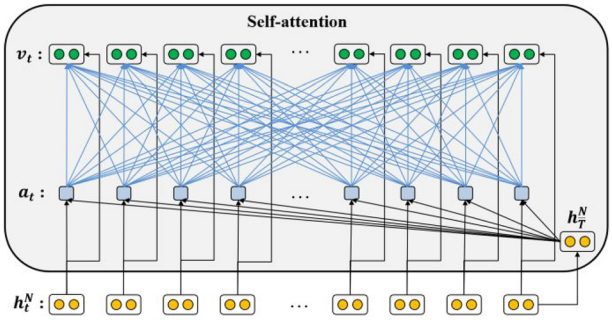

FIGURE 4 Self-attention diagram

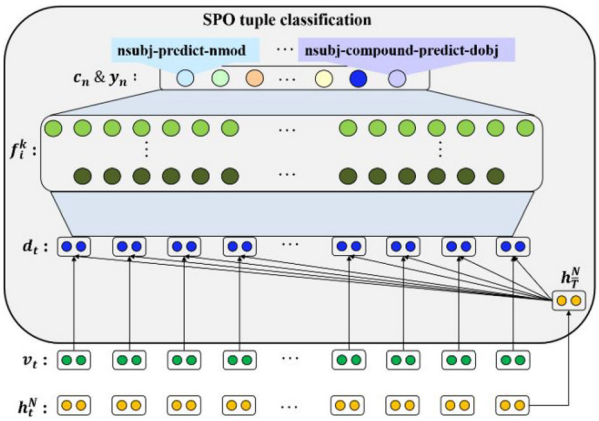

FIGURE 5 SPO tuple classification diagram

output step of M-Bi-LSTM to the self-attention values as follows:

$$\boldsymbol{d}_t = \Phi\left(\boldsymbol{v}_t, \boldsymbol{h}_{\overline{T}}^N\right). \tag{11}$$

where Φ is a combination function and $\boldsymbol{d}_t$ is self-decoding value at time t.

In encoder–decoder LSTM models, such as translation, the decoder maps the encoder's HS attention to the decoder's HS. However, the self-decoding maps the self-attention value ($\boldsymbol{v}_t$) to the encoder's HS ($\boldsymbol{h}_{\overline{T}}^N$).

Second, the results are given to a feedforward neural network (FFNN) using the self-decoding values as follows (12):

$$\boldsymbol{f}_i^k = \sigma^k\left(\sum_{j=1}^{J^k} \boldsymbol{W}_{i,j}^k \boldsymbol{f}_j^{k-1} + \boldsymbol{b}^k\right). \tag{12}$$

Here, $\boldsymbol{f}_i^0 = \boldsymbol{d}_i$ ($i = 1$ to T), $J^1 = T$, J is the vector size on the $(k-1)$th layer, σ^k is the activation function on the k^{th} layer, and k is the FFNN layer ($k = 1$ to L). Furthermore, $\boldsymbol{W}^k$ is the weight matrix mapping the $(k-1)$th layer to the k^{th} layer, and $\boldsymbol{b}^k$ is the bias of the k^{th} layer.

Third, the fully connected (FC) layer and softmax are performed for SPO tuple recognition, as shown in (13), (14), respectively:

$$c_n = \sigma\left(\sum_{m=1}^{M} \boldsymbol{W}_{n,m} \boldsymbol{f}_m^L + \boldsymbol{b}\right), \tag{13}$$

$$y_n = \frac{\exp(c_n)}{\sum_{j=1}^{N}\exp(c_j)}. \tag{14}$$

Here, $\boldsymbol{f}^L$ is the last layer of FFNN, c_n is the FC layer results ($n = 1$ to N), N is the number of classes, M is the

vector size on the last layer of FFNN, and y_n is the softmax value of c_n.

The SPO tuple of the NL sentence is recognized to extract knowledge through the end-to-end tuple recognition system using DG-M-Bi-LSTM. For example, the SPO tuple relation (nsubj-predict-dobj) can be recognized when an NL sentence (Google has announced their Android platform for mobile devices) is input into the DG-M-Bi-LSTM. Therefore, SPO tuple knowledge (Google–announce–Android platform) can be extracted using the SPO tuple relation and noun phrase chunk (Figure 2).

```
1:  loop (while EOF)
2:    read T_t & S_t
3:    generate D_t & L_t
4:    find T_t^p & T_t^s & T_t^o in L_t
5:    if ((n_t^p = 0) || (n_t^s = 0) || (n_t^0 = 0))
6:      continue
7:    select S_t^p & S_t^s & S_t^o
8:    if ((DD (S_t^p, S_t^s) > threshold) || (DD (S_t^p, S_t^o) > threshold))
9:      continue
10:   extract R_t
11:   generate G_t
12:   save T_t & S_t& G_t & R_t
13: end loop
```

FIGURE 6 Pseudo-code of DG-M-Bi-LSTM learning data generation

4 | EXPERIMENTAL RESULTS

The proposed DG-M-Bi-LSTM model is compared with the NL-based model and the recently proposed model in NLP studies to evaluate its effectiveness.

4.1 | Learning data

The 229 476 pairs were generated as learning data. The learning data for DG-M-BI-LSTM are pairs of dependency syntax analysis and SPO relations. To conduct a dependency syntax analysis and SPO tuple pairs, dependency parsing for each NL sentence was conducted, and SPO relations from DG parse tree were extracted. The detailed explanation of the learning data generation is as follows.

The open language learning for information extraction (OLLIE) provided 3 048 961 pairs of the SPO tuples and NL sentences [18]. However, there are many NL sentences where the SPO tuple is not the SPO relation. For example, the SPO tuple (Paypal–accept–Visa) is not the SPO relation of the NL sentence (We **accept Paypal**, Amex, Mastercard, and **Visa**). Therefore, 229 476 pairs were generated as the DG-M-Bi-LSTM learning data based on the OLLIE data (Figure 6). The SPO tuple (T_t) and NL sentence (S_t) are read from the OLLIE data. Then, the DG parse tree (D_t) of the NL sentence is generated, and the lemma of the NL sentence (L_t) is generated to process various types of words. The words that match the predict (T_t^p), subject (T_t^s), and object (T_t^o) in the SPO tuple are found in the lemma of the NL sentence. If the number of words matching the predict, subject, and object (n_t^p, n_t^s, and n_t^o, respectively) are all nonzero, the predict (S_t^p), subject (S_t^s), and object (S_t^o) are selected in the NL sentence. If the DG distance (DD) between the predict and subject and between the predict and object is both less than the threshold, the DG-based SPO tuple relation (R_t) is extracted from the DG parse tree. Then, the DG relation sequence (G_t) of the NL sentence is generated. Finally, the SPO tuple, NL sentence, DG relation sequence, and DG-based SPO tuple relation are stored in the DG-M-Bi-LSTM learning data.

There are 20 classes in 229 476 sentences. Table 1 presents the DG-based SPO tuple relations of the 20 classes.

Table 2 presents the selected DG relations in the DG-based SPO tuple relations.

4.2 | DG-based self-attention M-Bi-LSTM

To evaluate whether the proposed model performs well with a small amount of learning data, this study conducted training and evaluation tests using limited learning data (10 000) and all learning data (229 476). Training and evaluation tests were conducted using 70% and 30% of the learning data, respectively.

To confirm the accuracy of the SPO tuple recognition in the absence of self-attention M-Bi-LSTM, the SPO tuple recognition is estimated using only DG. The SPO tuple is recognized with DG embedding, two FFNN layers, and an FC layer. When all learning data are used, the SPO tuple recognition accuracy is 0.578, 0.838, and 0.907 for top-1, top-3, and top-5, respectively. When limited learning data are used, the SPO tuple recognition accuracy is 0.507, 0.736, and 0.827 for top-1, top-3, and top-5, respectively.

To determine the effect of multilayered and bidirectional DNN model on the performance of the SPO tuple recognition, SPO tuple recognitions are estimated by combining several layers ($l = 1$, 2, and 3) and directions [$d =$ unidirection (uni.) and bidirection (bi.)], as shown in Figure 7 and Table 3. The two-layered ($l = 2$) and bidirectional ($d =$ bi.) LSTM model outperforms other models in the SPO tuple recognition.

TABLE 1 DG-based SPO tuple relations of NL sentences

Classes	DG-based SPO tuple relations
Class 1	nsubj-predict-nmod
Class 2	nsubj-predict-dobj
Class 3	nsubj-predict-compound-nmod
Class 4	nmod-predict-nsubj
Class 5	nsubj-predict-nmod-dobj
Class 6	nsubj-predict-nmod-nmod
Class 7	nsubjpass-predict-nmod
Class 8	nsubj-predict-compound-dobj
Class 9	nsubj-predict-nsubj-ccomp
Class 10	nsubj-compound-predict-nmod
Class 11	nsubj-predict-nmod:tmod
Class 12	ccomp-nsubj-predict-nsubj
Class 13	dobj-predict-nsubj
Class 14	nsubj-predict-conj-nmod
Class 15	nmod-predict-nsubjpass
Class 16	dobj-predict-nmod
Class 17	nsubj-compound-predict-compound-dobj
Class 18	nsubj-conj-predict-nmod
Class 19	nsubj-predict-xcomp
Class 20	nsubj-compound-predict-dobj

TABLE 2 Selected DG relation features

DG relation	Description
nsubj	Nominal subject
nmod	Nominal modifier
dobj	Direct object
nsubjpass	Passive nominal subject
ccomp	Clausal complement
tmod	Temporal modifier
conj	Conjunct
xcomp	Open clausal complement

The DG-M-Bi-LSTM model consists of two LSTM layers, each with 768 hidden nodes, and two FFNN layers.

Figure 8 shows the evaluation test results using all learning data, showing the change in SPO tuple recognition accuracy at each epoch over 400 epochs. As the epochs increase, the SPO tuple recognition accuracy increases from 0.409 to 0.633 for top-1, 0.553 to 0.935 for top-3, and 0.659 to 0.972 for top-5 in the evaluation test.

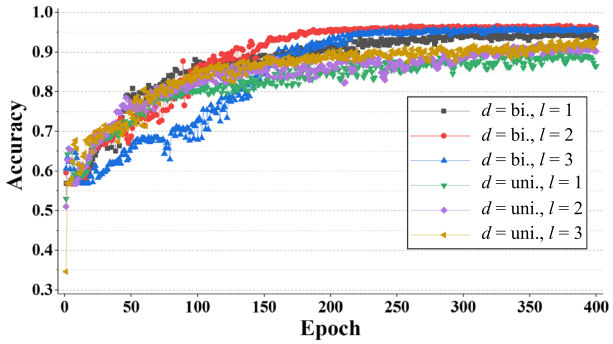

FIGURE 7 Ablation test results of DG-M-Bi-LSTM model: top-5 accuracy using limited learning data (10 000)

Figure 9 shows the evaluation test results using limited learning data. As the epochs increase, the SPO tuple recognition accuracy increases from 0.080 to 0.733 for top-1, 0.471 to 0.930 for top-3, and 0.596 to 0.968 for top-5 in the evaluation test.

The evaluation test was conducted four times, and the results are shown in Figure 10. Figure 10A,B shows the accuracy of the four evaluation tests (T1 through T4) and average for top-1, top-3, and top-5 when using limited learning data and all learning data, respectively.

When using limited learning data, the average accuracy is 0.697, 0.902, and 0.948 for top-1, top-3, and top-5, respectively. The standard deviation (SD) is 0.028, 0.021, and 0.018 for top-1, top-3, and top-5, respectively. When using all learning data, the average accuracy is 0.634, 0.937, and 0.974 for top-1, top-3, and top-5, respectively. The SD is 0.002, 0.003, and 0.001 for top-1, top-3, and top-5, respectively.

4.3 | NL-based self-attention M-Bi-LSTM

The accuracy using NL can be compared with the accuracy using DG by measuring the accuracy of the SPO tuple recognition of the self-attention M-Bi-LSTM model using NL instead of DG. NL-based self-attention M-Bi-LSTM (NL-M-Bi-LSTM) consists of two LSTM layers, each with 768 hidden nodes, and two FFNN layers. The evaluation test was performed four times, and the results are shown in Figure 11.

When using limited learning data, the average accuracy is 0.435, 0.743, and 0.869 for top-1, top-3, and top-5, respectively. The SD values are 0.013, 0.015, and 0.008 for top-1, top-3, and top-5, respectively. When using all learning data, the average accuracy is 0.476, 0.732, and 0.843 for top-1, top-3, and top-5, respectively. The SD values are 0.008, 0.009, and 0.004 for top-1, top-3, and top-5, respectively.

TABLE 3 Ablation test results of DG-M-Bi-LSTM model

	Limited learning data			All learning data		
Model	**top-1**	**top-3**	**top-5**	**top-1**	**top-3**	**top-5**
d = bi., l = 1	0.709	0.912	0.948	0.627	0.928	0.970
d = bi., l = 2	**0.733**	**0.930**	**0.968**	**0.633**	**0.935**	**0.972**
d = bi., l = 3	0.719	0.927	0.962	0.631	0.931	0.969
d = uni., l = 1	0.585	0.836	0.899	0.566	0.825	0.906
d = uni., l = 2	0.612	0.847	0.921	0.585	0.861	0.931
d = uni., l = 3	0.657	0.879	0.930	0.597	0.891	0.949

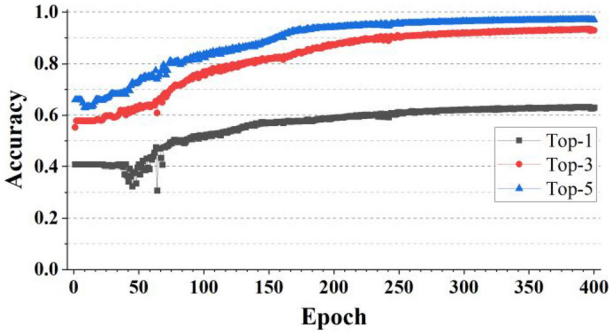

FIGURE 8 DG-M-Bi-LSTM evaluation test results using all learning data (229 476)

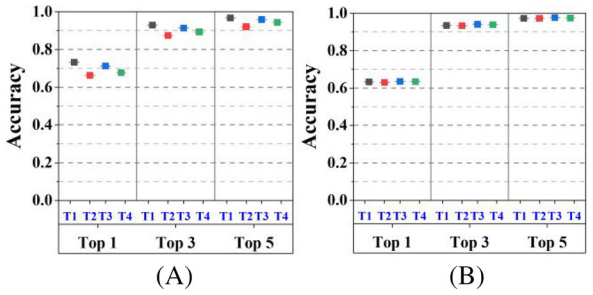

FIGURE 10 SPO tuple recognition accuracy of DG-M-Bi-LSTM when using (A) limited learning data (10 000) and (B) all learning data (229 476)

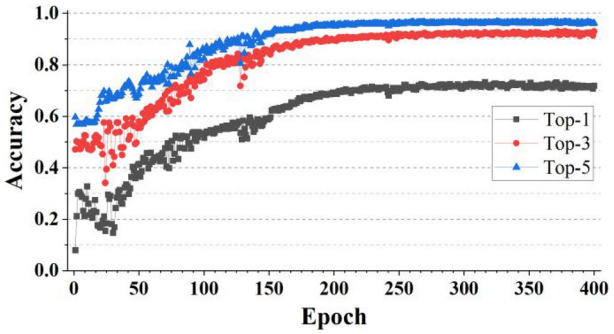

FIGURE 9 DG-M-Bi-LSTM evaluation test results using limited learning data (10 000)

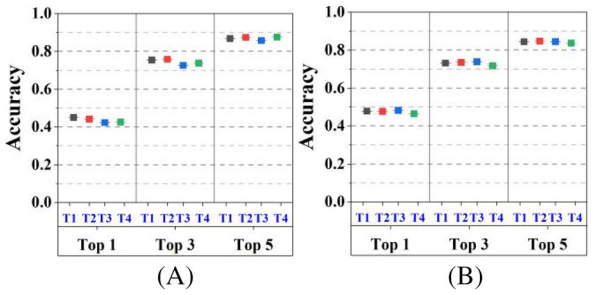

FIGURE 11 SPO tuple recognition accuracy of NL-M-Bi-LSTM when using (A) limited learning data (10 000) and (B) all learning data (229 476)

4.4 | NL-based BERT

The accuracy according to the DNN model using NL can be compared by measuring the accuracy of the SPO tuple recognition of the BERT model instead of the self-attention M-Bi-LSTM model using NL.

The pretrained BERT consists of 24 layers, each with 1024 hidden nodes, and 16 self-attention heads. After fine-tuning the BERT parameters using 70% of the learning data, an evaluation test was conducted using the remaining 30%. The NL-based BERT (NL-BERT) evaluation test was conducted four times, and the results are shown in Figure 12.

When using limited learning data, the average accuracy is 0.312, 0.575, and 0.638 for top-1, top-3, and top-5, respectively. The SD values are 0.123, 0.071, and 0.074 for top-1, top-3, and top-5, respectively. When using all learning data, the average accuracy is 0.551, 0.684, and 0.752 for top-1, top-3, and top-5, respectively. The SD values are 0.142, 0.226, and 0.190 for top-1, top-3, and top-5, respectively.

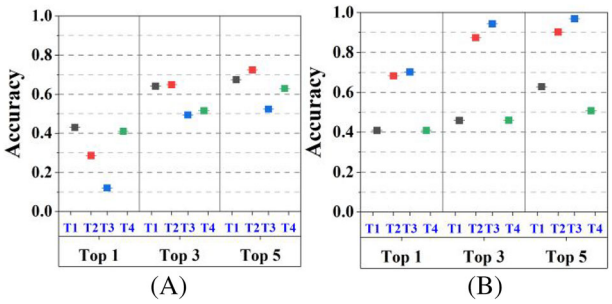

FIGURE 12 SPO tuple recognition accuracy of NL-BERT when using (A) limited learning data (10 000) and (B) all learning data (229 476)

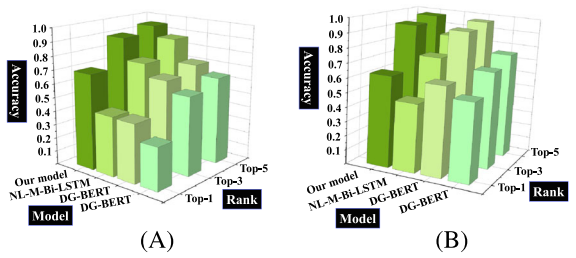

FIGURE 14 Comparison of SPO tuple recognition accuracy results when using (A) limited learning data (10 000) and (B) all learning data (229 476)

4.5 | DG-based BERT

The accuracy according to the DNN model using DG can be compared by measuring the SPO tuple recognition accuracy of the BERT model using DG. The DG-based BERT (DG-BERT) evaluation test was conducted four times, and the results are shown in Figure 13.

When using limited learning data, the average accuracy is 0.428, 0.653, and 0.705 for top-1, top-3, and top-5, respectively. The SD values are 0.004, 0.077, and 0.061 for top-1, top-3, and top-5, respectively. When using all learning data, the average accuracy is 0.625, 0.934, and 0.966 for top-1, top-3, and top-5, respectively. The SD values are 0.004, 0.004, and 0.002 for top-1, top-3, and top-5, respectively.

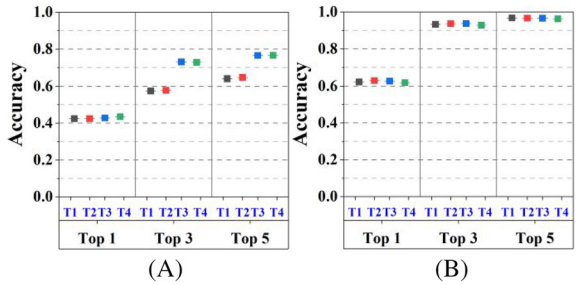

FIGURE 13 SPO tuple recognition accuracy of DG-BERT when using (A) limited learning data (10 000) and (B) all learning data (229 476)

4.6 | Comparison of the results

The proposed DG-M-Bi-LSTM achieves better accuracy than NL-BERT, DG-BERT, and NL-M-Bi-LSTM, as shown in Table 4 and Figure 14.

The experimental results show that the proposed DG-M-Bi-LSTM model achieves the best performance at recognizing SPO tuples in NL sentences even if it has fewer DNN parameters than BERT. The BERT consists of 24 layers, 1024 hidden sizes, and 16 self-attention heads containing 340 M parameters [30]. However, the proposed model consists of two LSTM layers, 768 LSTM-hidden sizes, and two FFNN layers containing only 15 M parameters. Because the proposed model requires a smaller parameter size than BERT, it can be used in various applications that require less memory or learning time than BERT. For instance, it shows superior performance compared to BERT when few learning data are used.

TABLE 4 Comparison of the SPO tuple recognition accuracy results using DG-M-Bi-LSTM, NL-M-Bi-LSTM, DG-BERT, and NL-BERT

	Limited learning data			All learning data		
Model	**top-1**	**top-3**	**top-5**	**top-1**	**top-3**	**top-5**
NL-BERT	0.312	0.575	0.638	0.551	0.684	0.752
DG-BERT	0.428	0.653	0.705	0.625	0.934	0.966
NL-M-Bi-LSTM	0.435	0.743	0.869	0.476	0.732	0.843
DG-M-Bi-LSTM (our model)	**0.697**	**0.902**	**0.948**	**0.634**	**0.937**	**0.974**

5 | CONCLUSION

This study addressed an SPO tuple recognition DNN model based on DG to autonomously extract knowledge from NL sentences. The DG-M-Bi-LSTM model consists of DG embedding, M-Bi-LSTM, self-attention, and SPO tuple classification. The DG embedding performs DG parsing for NL sentences and embedding for the generated DG relations. The M-Bi-LSTM model receives the DG embeddings as input vectors and performs deep learning using M-Bi-LSTM. Self-attention calculates the attention scores for each step of M-Bi-LSTM and generates attention values. The SPO tuple classification determines the SPO tuple relations of the NL sentences.

Furthermore, the proposed DG-M-Bi-LSTM model was compared with NL-BERT, DG-BERT, and NL-M-Bi-LSTM to evaluate its effectiveness. The experimental results show that DG-M-Bi-LSTM achieves better results in terms of SPO tuple recognition accuracy than other compared models, even though the proposed model has fewer DNN parameters than BERT. Unlike BERT, it shows good accuracy even with limited learning data. Therefore, the proposed model can be used even with limited learning data. Additionally, the pretrained parameter can be applied to different domains because DG-M-Bi-LSTM learns the structural relations in NL sentences instead of learning the semantic relations of specific words.

This study recognizes the SPO tuple from an NL sentence using a DG-based DNN model. However, the SPO tuple can be extracted using several NL sentences in a paragraph. Therefore, in future studies, research on recognizing SPO tuple in a paragraph is required to infer SPO tuples from multiple NL sentences.

ACKNOWLEDGEMENTS

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government (21ZS1100, Core Technology Research for Self-Improving Integrated Artificial Intelligence System).

ORCID

Joon-young Jung https://orcid.org/0000-0001-6964-4005

REFERENCES

1. S. Auer et al., *DBpedia: A nucleus for a web of open data*, in Proc. Int. Semantic Web Conf. (Busan, Republic of Korea), Nov. 2007, pp. 722–735.
2. D. Vrandečić and M. Krötzsch, *Wikidata: A free collaborative knowledgebase*, Commun. ACM **57** (2014), no. 10, 78–85.
3. F. M. Suchanek, G. Kasneci, and G. Weikum, *YAGO: A core of semantic knowledge*, in Proc. Int. Conf. WWW (Banff, Canada), May 2007, pp. 697–706.
4. N. Kolitsas, O.-E. Ganea, and T. Hofmann, *End-to-end neural entity linking*, in Proc. Conf. Comput. Nat. Lang. Learn. (Brussels, Belgium), Aug. 2018, pp. 519–529.
5. B. D. Trisedya, J. Qi, and R. Zhang, *Entity alignment between knowledge graphs using attribute embeddings*, in Proc. AAAI Conf. on Artif. Intell. (Honolulu, HI, USA), July 2019, pp. 297–304.
6. B. D. Trisedya, J. Qi, R. Zhang, and W. Wang, *GTR-LSTM: A triple encoder for sentence generation from RDF data*, in Proc. Annu. Meet. Assoc. Comput. Linguistics (Melbourne, Australia), July 2018, pp. 1627–1637.
7. M. J. Cafarella et al., *WebTables: Exploring the power of tables on the web*, in Proc. Very Large Data Base Endowment (Auckland, New Zealand), Aug. 2008, pp. 538–549.
8. O. Lehmberg et al., *A large public corpus of web tables containing time and context metadata*, in Proc. Int. Conf. Companion WWW (Montréal, Canada), Apr. 2016, pp. 75–76.
9. B. Fetahu, A. Anand, and M. Koutraki, *TableNet: An approach for determining fine-grained relations for wikipedia tables*, in Proc. Int WWW Conf. (San Francisco, CA, USA), May 2019, pp. 2736–2742.
10. M. Mintz et al., *Distant supervision for relation extraction without labeled data*, in Proc. Joint Conf. Assoc. Comput. Linguistics & Int. Joint Conf. Natural Lang. Process. AFNLP (Suntec, Singapore), Aug. 2009, pp. 1003–1011.
11. S. Riedel, L. Yao, and A. McCallum, *Modeling relations and their mentions without labeled text*, in Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases (Barcelona, Spain), Sept. 2010, pp. 148–163.
12. S. Riedel et al., *Relation extraction with matrix factorization and universal schemas*, in Proc. N. Am. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol. (Atlanta, GA, USA), June 2013, pp. 74–84.
13. D. Zeng et al., *Distant supervision for relation extraction via piecewise convolutional neural networks*, in Proc. Conf. Empir. Methods Nat. Lang. Process. (Lisbon, Portugal), Sept. 2015, pp. 1753–1762.
14. Y. Lin et al., *Neural relation extraction with selective attention over instances*, in Proc. Annu. Meet. Assoc. Comput. Linguistics (Berlin, Germany), Aug. 2016, pp. 2124–2133.
15. P. Zhou et al., *Distant supervision for relation extraction with hierarchical selective attention*, Neural Netw. **108** (2018), 240–247.
16. M. Banko et al., *Open information extraction from the web*, in Proc. Int. Joint Conf. Artif. Intell. (Hyderabad, India), Jan. 2007, pp. 2670–2676.
17. A. Fader, S. Soderland, and O. Etzioni, *Identifying relations for open information extraction*, in Proc. Conf. Empir. Methods Nat. Lang. Process. (Edinburgh, UK), July 2011, pp. 1535–1545.
18. M. Schmitz et al., *Open language learning for information extraction*, in Proc. Conf. Empir. Methods Nat. Lang. Process. (Jeju, Republic of Korea), July 2012, pp. 523–534.
19. L. D. Corro and R. Gemulla, *ClausIE: Clause-based open information extraction*, in Proc. Int. Conf. WWW (Rio de Janeiro, Brazil), May 2013, pp. 355–366.
20. K. Gashteovski, R. Gemulla, and L. D. Corro, *MinIE: Minimizing facts in open information extraction*, in Proc. Conf. Empir. Methods Nat. Lang. Process. (Copenhagen, Denmark), Sept. 2017, pp. 2630–2640.

21. G. Angeli, M. J. J. Premkumar, and C. D. Manning, *Leveraging linguistic structure for open domain information extraction*, in Proc. Assoc. Comput. (Beijing, China), July 2015, pp. 344–354.
22. G. Stanovsky et al., *Supervised open information extraction*, in Proc. N. Am. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol. (New Orleans, LA, USA), June 2018, pp. 885–895.
23. L. Cui, F. Wei, and M. Zhou, *Neural open information extraction*, in Proc. Annu. Meet. Assoc. Comput. Linguistics (Melbourne, Australia), May 2018, pp. 407–413.
24. M. Sun et al., *Logician: A unified end-to-end neural approach for open-domain information extraction*, in Proc. Web Search Data Min. (Los Angeles, CA, USA), Feb. 2018, pp. 556–564.
25. S. Jia, Y. Xiang, and X. Chen, *Supervised neural models revitalize the open relation extraction*, arXiv preprint, CoRR, 2018, arXiv: 1809.09408.
26. Z. Jiang, P. Yin, and G. Neubig, *Improving open information extraction via iterative rank-aware learning*, in Proc. Annu. Meet. Assoc. Comput. Linguistics (Florence, Italy), May 2019, pp. 5295–5300.
27. B. D. Trisedya et al., *Neural relation extraction for knowledge base enrichment*, in Proc. Annu. Meet. Assoc. Comput. Linguistics (Florence, Italy), July 2019, pp. 229–240.
28. P. Shi and J. Lin, *Simple BERT models for relation extraction and semantic role labeling*, arXiv preprint, CoRR, 2019, arXiv: 1904.05255.
29. Y. Papanikolaou, I. Roberts, and A. Pierleoni, *Deep bidirectional transformers for relation extraction without supervision*, arXiv preprint, CoRR, 2019, arXiv: 1911.00313.
30. J. Devlin et al., *BERT: Pre-training of deep bidirectional transformers for language understanding*, in Proc. N. Am. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol. (Minneapolis, MN, USA), May 2019, pp. 4171–4186.
31. M. D. Marneffe et al., *Universal Stanford dependencies: A cross-linguistic typology*, in Proc. Int. Conf. Lang. Resour. Eval. (Reykjavik, Iceland), May 2014, pp. 4585–4592.
32. J. Nivre et al., *Universal dependencies v1: A multilingual treebank collection*, in Proc. Int. Conf. Lang. Resour. Eval. (Portorož, Slovenia), May 2016, pp. 1659–1666.
33. N. Nakashole, G. Weikum, and F. Suchanek, *PATTY: A taxonomy of relational patterns with semantic types*, in Proc. Conf. Empir. Methods Nat. Lang. Process. (Jeju, Republic of Korea), July 2012, pp. 1135–1145.
34. D. Klein and C. D. Manning, *Accurate unlexicalized parsing*, in Proc. Annu. Meet. Assoc. Comput. Linguistics (Sapporo, Japan), **34** (2003), pp. 423–430.
35. D. Chen and C. D. Manning, *A fast and accurate dependency parser using neural networks*, in Proc. Conf. Empir. Methods Nat. Lang. Process. (Doha, Qatar), Oct. 2014, pp. 740–750.
36. T. Mikolov et al., *Efficient estimation of word representations in vector space*, arXiv preprint, CoRR, 2013, arXiv: 1301.3781.
37. T. Mikolov et al., *Distributed representations of words and phrases and their compositionality*, in Proc. Neural Inf. Process. Syst. (Lake Tahoe, NV, USA), Dec. 2013, pp. 3111–3119.
38. J. Pennington, R. Socher, and C. D. Manning, *GloVe: Global vectors for word representation*, in Proc. Conf. Empir. Methods Nat. Lang. Process. (Doha, Qatar), Oct. 2014, pp. 1532–1543.
39. R. Jozefowicz, W. Zaremba, and I. Sutskever, *An empirical exploration of recurrent network architectures*, in Proc. Int. Conf. Mach. Learn. (Lille, France), June 2015, pp. 2342–2350.
40. M. F. Y. Ghadikolaie, E. Kabir, and F. Razzazi, *Sub-word based offline handwritten farsi word recognition using recurrent neural network*, ETRI J. **38** (2016), no. 4, 703–713.
41. W. Khan et al., *Deep recurrent neural networks with word embeddings for Urdu named entity recognition*, ETRI J. **42** (2020), no. 1, 90–100.
42. Y. Bengio, P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult*, IEEE Trans. Neural Netw. **5** (1994), no. 2, 157–166.
43. S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Comput. **9** (1997), no. 8, 1735–1780.
44. F. A. Gers, J. Schmidhuber, and F. Cummins, *Learning to forget: Continual prediction with LSTM*, in Proc. Int. Conf. Artif. Neural Netw. (Edinburgh, UK), Oct. 1999, pp. 850–855.
45. F. A. Gers and J. Schmidhuber, *Recurrent nets that time and count*, in Proc. Int. Joint Conf. Neural Netw. (Como, Italy), July 2000, pp. 189–194.
46. A. Graves and J. Schmidhuber, *Framewise phoneme classification with bidirectional LSTM and other neural network architectures*, Neural Netw. **18** (2005), no. 5–6, 602–610.
47. M. Schuster and K. K. Paliwal, *Bidirectional recurrent neural networks*, IEEE Trans. Signal Process. **45** (1997), no. 11, 2673–2681.
48. A. Graves, N. Jaitly, and A. Mohamed, *Hybrid speech recognition with deep bidirectional LSTM*, in Proc. IEEE Workshop Autom. Speech Recognit. Underst. (Olomouc, Czech Republic), Dec. 2013, pp. 273–278.
49. F. U. M. Ullah et al., *Short-term prediction of residential power energy consumption via CNN and multilayer bi-directional LSTM Networks*, IEEE Access **8** (2019), 123369–123380.
50. V. Mnih et al., *Recurrent models of visual attention*, in Proc. Int. Conf. Neural Inf. Process. Syst. (Montreal, Canada), Dec. 2014, pp. 2204–2212.
51. D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, arXiv preprint, CoRR, 2014, arXiv: 1409.0473.
52. A. M. Rush, S. Chopra, and J. Weston, *A neural attention model for abstractive sentence summarization*, in Proc. Empir. Methods Nat. Lang. Process. (Lisbon, Portugal), Sept. 2015, pp. 379–389.
53. Z. Zhang, Y. Zou, and C. Gan, *Textual sentiment analysis via three different attention convolutional neural networks and cross-modality consistent regression*, Neurocomputing **275** (2018), 1407–1415.
54. G. Liu and J. Guo, *Bidirectional LSTM with attention mechanism and convolutional layer for text classification*, Neurocomputing **337** (2019), 325–338.
55. M. P. Akhter et al., *Document-level text classification using single-layer multisize filters convolutional neural network*, IEEE Access **8** (2020), 42689–42707.
56. A. Vaswani et al., *Attention is all you need*, in Proc. Conf. Neural Inf. Process. Syst. (Long Beach, CA, USA), Dec. 2017, pp. 6000–6010.
57. J. Cheng, L. Dong, and M. Lapata, *Long short-term memory-networks for machine reading*, in Proc. Conf. Empir. Methods Nat. Lang. Process. (Austin, TX, USA), Nov. 2016, pp. 551–561.
58. A. Parikh et al., *A decomposable attention model for natural language inference*, in Proc. Empir. Methods Nat. Lang. Process. (Austin, TX, USA), Nov. 2016, pp. 2249–2255.

AUTHOR BIOGRAPHY

Joon-young Jung received his BS and MS degrees in computer network engineering from Soongsil University, Seoul, South Korea, in 1996 and 2000, respectively, and his PhD degree in information communications engineering from Chungnam National University, Daejeon, South Korea, in 2015. Since 2000, he has been a research engineer at Electronics and Telecommunications Research Institute, Daejeon, South Korea. His main research interests are natural language processing and visual intelligence.

How to cite this article: J. Jung, *DG-based SPO tuple recognition using self-attention M-Bi-LSTM*, ETRI Journal (2021), 1–12. https://doi.org/10.4218/etrij.2020-0460