Journal of Multimedia Information System VOL. 9, NO. 2, June 2022 (pp. 93-102): ISSN 2383-7632 (Online)
https://doi.org/10.33851/JMIS.2022.9.2.93

Proper Noun Embedding Model for the Korean Dependency Parsing

Gyu-Hyeon Nam[1], Hyun-Young Lee[2], Seung-Shik Kang[3*]

Abstract

Dependency parsing is a decision problem of the syntactic relation between words in a sentence. Recently, deep learning models are used for dependency parsing based on the word representations in a continuous vector space. However, it causes a mislabeled tagging problem for the proper nouns that rarely appear in the training corpus because it is difficult to express out-of-vocabulary (OOV) words in a continuous vector space. To solve the OOV problem in dependency parsing, we explored the proper noun embedding method according to the embedding unit. Before representing words in a continuous vector space, we replace the proper nouns with a special token and train them for the contextual features by using the multi-layer bidirectional LSTM. Two models of the syllable-based and morpheme-based unit are proposed for proper noun embedding and the performance of the dependency parsing is more improved in the ensemble model than each syllable and morpheme embedding model. The experimental results showed that our ensemble model improved 1.69%p in UAS and 2.17%p in LAS than the same arc-eager approach-based Malt parser.

Key Words: Dependency Parsing, LSTM, Proper Noun Embedding, Malt Parser, Transition-Based Model.

I. INTRODUCTION

Dependency parsing is to find a governor of each word and its dependency relation between the governor and the dependent. The parser analyzes the syntactic structure of a sentence by predicting the dependency relation between the words in a sentence [1-2]. The dependency relation in a sentence helps understand the meaning of the sentence. Thus, dependency parsing plays an important role in natural language processing. Recently, the deep learning model has made remarkable improvements in signal processing, computer vision, and natural language processing. In the deep learning model for natural language processing tasks, words and sentences are represented in a continuous vector space to be used as input features [3-10]. However, proper nouns in the input sentence make it difficult to represent a proper noun in the continuous vector space [11-12].

Dependency parsing techniques are categorized into cascaded chunking, graph-based, and transition-based approaches. When comparing the time complexity of each technique, the cascaded chunking and graph-based approach have the time complexity of $O(n^2)$ and $O(n^3)$ respectively, The time complexity of transition-based parsing, however, is $O(n)$, namely the transition-based approach is more effective in the time complexity than other methods [13-17]. Transition-based dependency parsing predicts a correct transition based on features extracted from the configuration, which consists of a stack, a buffer, and a set of dependency arc, and updates the configuration by taking actions according to the features extracted from the configuration. When the state of configuration reaches a terminal condition, the dependency parsing tree is completed according to the dependency relation and the relation type, where the deep learning model is used to determine the actions from features extracted from the configuration [18-20].

In early studies of dependency parsing with deep learning model, the features extracted from the configuration are trained by feed-forward neural network. The features consist of only the upper components of the stack and the buffer. Therefore, this neural network has the disadvantage that it can learn only from the features extracted from limited components. Lee et al. (2014) used NNLM (Neural Network Language Model) and word2vec to embed Korean morphemes, and it is trained by using feed-forward neural network with ReLU function and dropout method [21]. The studies based on recurrent neural network are conducted to utilize components in the configuration with LSTM cell to

Manuscript received March 26, 2022; Revised May 1, 2022; Accepted May 24, 2022. (ID No. JMIS-22M-03-011)
Corresponding Author (*): Seung-Shik Kang, +82-2-910-4800, sskang@kookmin.ac.kr
[1]DeepBrain AI, Seoul, Korea, ngh3053@deepbrainai.io
[2]KT Corporation, Seoul, Korea, lee.hyunyoung@kt.com
[3]Department of Artificial Intelligence, Kookmin University, Seoul, Korea, sskang@kookmin.ac.kr

learn long dependency [22-23]. Li and Lee (2015) proposed a recurrent neural network to solve the limited learning of feed-forward neural networks and they used LSTM to solve a vanishing gradient problem of recurrent neural network [24]. Na et al. (2016) suggested a learning model with stack-LSTM instead of learning configuration features at once [25-26]. In addition, there are researches combining the transition-based method and the graph-based method or other deep learning models such as seq2seq and SyntaxNet [27-29].

In case of the Korean sentence, a head is located on the right side of the dependent because the sentence structure follows a head final rule. If we parse a Korean sentence with traditional transition-based approach in reverse order, the direction of the relation ends up with a left-arc according to the head final rule. In order to improve the efficiency of the oracle, it is necessary to reduce the number of transitions. Lim et al. (2014) investigated to achieve it by using the head final rules in the Korean language to run the transition-based parser in reverse order [30]. Although they improve the efficiency of a parser, they do not resolve OOV problem to deal with proper noun that hardly occurs in train corpus. In this paper, we propose a Korean dependency parsing method to represent proper noun in a continuous vector space by replacing proper noun with a special token. We employ one of the transition-based approaches, arc-eager approach, with neural network, and the neural network-based model was constructed by multi-layer bidirectional LSTM.

II. TRANSITION-BASED PARSING

2.1. Transition-based Dependency Parsing

Transition-based dependency parsing predicts a correct transition between words in a sentence with features from the configuration and then derives a target dependency parse tree. The transition-based dependency parsing model is implemented by arc-standard and arc-eager approach [31]. The arc-standard approach asserts the relation between two elements at the top of the stack. If a relation exists, it determines the direction and the dependency label of the arc. If the relation does not exist, it takes one word from the front of the input buffer onto the stack, denoted as shift operator. The initial state of the configuration for parsing a sentence consists of a stack with a root token and a buffer with all the words in the sentence. The root token means the top-level head of the sentence. The arc-standard approach terminates when the stack contains a single token (root token) and the buffer is empty.

The arc-eager approach, on the other hand, asserts a relation between the word at the top of the stack and the word at the front of input buffer. If the relation exists, unlike the arc-standard, for the directed arc, if the word at the top of the stack is a head, push the word at the front of input buffer onto the stack without removing the top word of the stack (right-arc). If the word at the front of input buffer is a head, then the word at the top of the stack is removed (left-arc). The arc-eager approach has the initial configuration for parsing a sentence, which is the empty stack and the input buffer containing input words. The arc-eager approach is terminated if it is any configuration with the empty buffer and, in addition to shift operator, has one more operator, denoted as reduce operator which pops the top element from the stack. For our experiment of transition-based dependency parsing for a Korean sentence, we exploited the arc-eager approach. Although the arc-eager approach has one more operation rather than the arc-standard approach, the reason to use the arc-eager approach is it relatively terminates well than the arc-standard approach for the terminal condition of the arc-eager approach even if an error behavior was included.

2.2. Korean Dependency Parsing Algorithm

For the Korean dependency parsing, an input sentence is expressed into the configuration consisting of a stack, a buffer, and previous actions. The input sentence is set in a reverse order at the initial configuration and then the parser starts predicting the correct transition on the feature ex-

Table 1. Korean dependency parsing algorithm.

Algorithm: Korean dependency parsing

```
Input: a sequence of words in a sentence
Output: a list of governors and dependency relations

stack, buffer ← make_stack(), make_buffer(eojeols)
action ← make_list()
buffer ← reverse(buffer)

pred_head ← make_list(size=len(eojeols) + 1)
pred_rel ← make_list(size=len(eojeols) + 1)

for i in range(len(eojeols) * 2)
        if is_empty(buffer) then
                end algorithm
        next_action, relation ← oracle(stack, buffer, action)
        if next_action = 'RIGHT-ARC' then
                action.push(next_action)
        else if next_action = 'REDUCE' then
                action.push(next_action)
                pred_head[buffer.top()] ← stack.top()
                pred_rel[buffer.top()] ← relation
                stack.push(buffer.pop())
        else
                raise parse_error
end for
return pred_head, pred_rel
```

tracted from the current configuration. In Table 1, the algorithm shows the brief operation of the Korean dependency parsing.

A Korean sentence follows the head final rule. So, the head is always on the right side of a sentence. The directed arc in Korean sentences is left-arc in transition-based approach. However, there is a right-arc transition between the root of the sentence and the root token. Direction of the arc for Korean dependency parsing is right-arc where a buffer is initialized by pushing a sequence of words in reverse order. Parsing the sequence of words in reverse order has an advantage to take the right-arc operator of arc operators (i.e. both left-arc and right-arc) into account, it indicates our dependency parser computes the reduced number of the transition operation rather than considering both left-arc and right-arc. In our approach, since the direction of the arc for parsing a Korean sentence is only the right-arc, the stack has words, which are asserted as head in dependency parsing, and the buffer has words which are not asserted as head. The shift operator is not used since the word at the front of input buffer is always pushed into stack with right-arc operator. Thus, the feasible actions for Korean dependency parsing in reverse order is reduce operator and right-arc operator.

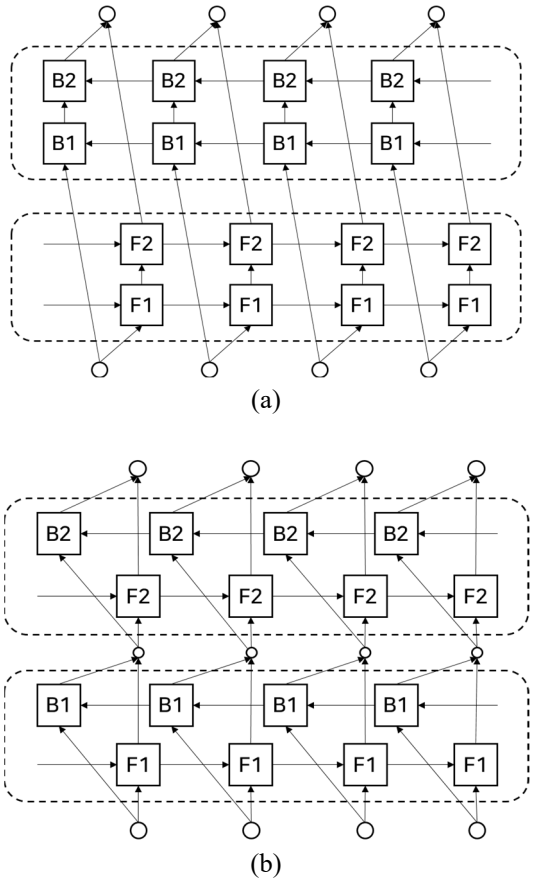

Fig. 1. Two models of multi-layer bidirectional LSTM.

2.3. Deep Learning Model Architecture

The bidirectional LSTM is used to extract feature to take into account the context on both the left-to-right and right-to-left direction. The multi-layer bidirectional LSTM calculates a context-dependent representation for each input using a bidirectional LSTM, and then use this representation as input to another bidirectional LSTM. The multi-layer structure makes the model to learn more complex features. The method to deliver the context-dependent representation to an upper layer can be split into two composition architectures as shown in Fig. 1. For example, with two-layers bidirectional LSTM, one composition architecture Fig. 1(a) is that the context-dependent information from the bottom layer is delivered in each forward and backward direction separately, it doesn't truly take into account the bidirectional representation between layers [6].

We propose another compositional architecture Fig. 1(b). It combines each context-dependent representation from left-to-right and right-to-left to make the model extract feature, which is dependent to bidirectional representation before moving it to the upper layer in multi-layer bidirectional LSTM such as Fig. 1(b). In the experiment, we compare two compositional architectures of multi-layer bidirectional LSTM without preprocessing proper noun to choose the appropriate compositional architecture for proper noun embedding model of Korean dependency parsing. The chosen composition architecture of multi-layer bidirectional LSTM is used to gain features from configuration for our proper noun embedding in Korean dependency parsing. The chosen composition architecture of multi-layer bidirectional LSTM is used to gain features from the configuration for our proper noun embedding in Korean dependency parsing.

Following the arc-eager transition-based approach and the head final rule in Korean, we set a stack, an input buffer, and the previous actions as configuration feature. We make our multi-layer bidirectional LSTM model to read the stack, the previous actions, and the input buffer initialized by pushing a sequence of words in reverse order. In our experiment, we make the oracle assert reduce operator and right-arc operation exclusive of left-arc operator and shift operator in the arc-eager transition-based approach.

The next action is predicted by input token, stack token, and previous actions as shown in Fig. 2. Extracting the features, the bidirectional LSTM is applied to each configuration (i.e. stack, buffer, and previous actions). As shown in Equation (1), The forward LSTM $\vec{h}_{config}$ reads the configuration feature left-to-right, and then the backward LSTM $\overleftarrow{h}_{config}$ reads right-to-left. Each context-dependent representation extracted from the forward and backward LSTM is added to embed each configuration into continuous vector space. In order to choose the transition for parsing the Korean sentence, we train our model to minimize the cross

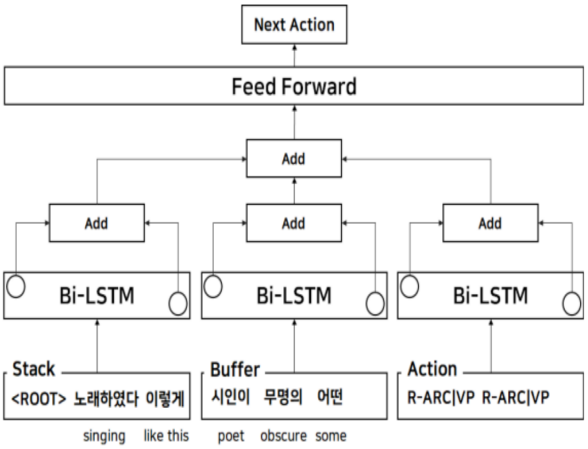

Fig. 2. Bi-directional LSTM model architecture for transition-based dependency parsing.

entropy loss between the action obtained by the oracle and the ground truth action.

$$\vec{h}_{config} = LSTM_{forward}\left(E\left(x_{config}\right)\right).$$
$$\overleftarrow{h}_{config} = LSTM_{backward}\left(E\left(x_{config}\right)\right).$$
$$h_{config} = \vec{h}_{config} + \overleftarrow{h}_{config}. \quad ((1)$$

III. PROPER NOUN EMBEDDING

Proper noun, represented in continuous vector space, causes an out-of-vocabulary problem. In particular, for proper nouns in that the proper noun appearing in test corpus is guaranteed to be incorporated in train corpus. The proper noun has a property that it is primarily concatenated with a particle. The particles grammatically and semantically play an important role in Korean to understanding the meaning of a sentence. In other words, when interpreting the meaning of a sentence and the relation of words in a sentence, the particle concatenated with proper noun is more significant part than proper noun itself. For identifying the relation of words, the proper noun embedding is not useful in dealing with the new proper noun unseen in training but the particles can be reused in such a case that what role (e.g. subject or object in a sentence) the unseen proper noun plays on the sentence.

Allowing this point, we propose proper noun representation method that we call proper noun embedding in our experiment. When proper noun representation is projected into continuous vector space, we adopt method to retain particle information and replace proper noun with a special token as in making the known token vector as preprocessing. In our experiment, we implement proper noun embedding in two embedding unit such as syllable and morpheme embedding unit. For each embedding unit, we split a word into each unit and then each unit vector is added onto word-level. For example, in the case of syllable embedding unit, we split a word into syllable unit and then the syllable embedding is added on word-level.

Specifically, two methods of proper noun embedding are possible. The first one is based on the ‘syllable’ unit and the second one is based on the ‘morpheme’ unit. In a sentence “서울에 갔다(went to Seoul)”, there are two types of tokenization (e.g. syllable and morpheme token) for the two words in the sentence “서울에(Seoul-E) 갔다(went)”. In the syllable-based embedding method, all the syllables in proper nouns are replaced by a special token in which embedding unit is a syllable. The proper noun ‘서울(Seoul)’ is further tokenized into ‘서(Seo)’ + ‘울(ul)’, and then the syllables of proper nouns are replaced by a special token such as ‘♬’. In the same way, ‘갔다(went)’ is tokenized into two syllables ‘갔(went)’ and ‘다(functional morpheme of verb ending)’ and then the syllable embedding is added onto word-level.

As for the morpheme-based embedding, morphological analyzer is used to extract morpheme tokens. Morphological analyzer tokenizes a sentence into a sequence of morphemes. An example of morpheme-based embedding in which proper noun ‘서울(Seoul)’ itself is replaced by a special token such as ‘♪’. In such a morpheme case, embedding unit is a morpheme. It shows embedding units of a morpheme sequence “서울(Seoul)+에(locative case particle)+가(go)+았(past particle in verb)+다(functional morpheme of verb ending)” for a sentence “서울에(Seoul-E) 갔다(went)”. The first word ‘서울에(Seoul-E)’ is tokenized into ‘서울(Seoul)’ + ‘에(locative case particle)’ and the second word ‘갔다(went)’ is tokenized into ‘가(go)’ + ‘았(past particle)’ + ‘다(verb ending)’. The difference of the syllable-based and morpheme-based method for proper noun embedding is whether to decompose a proper noun into smaller unit (e.g. syllable unit) or not. In other words, for any syllable (Korean character), proper noun embedding is the composition of syllable embedding in the proper noun. As for morpheme token, the proper noun embedding itself is projected in a vector continuous space.

We, furthermore, can attach a part-of-speech tag to a morpheme, and a word is made up of a combination of several morphemes with corresponding part-of-speech tags. The number of parts-of-speech tags combination increases according to the morphemes in a word and the part-of-speech vector is generated by adding all the morpheme parts that make up a word.

IV. EXPERIMENTS AND RESULTS

In this section, we describe our experiments and results

of model with and without the proposed proper noun embedding on dependency parsing task as follows. In Section 4.1, we explain dataset for Korean dependency parsing task. In order to select the model for proper noun embedding, we show the results of model without the proposed proper noun embedding in Section 4.2 and then, in Section 4.3, explore the performance with the proposed proper noun embedding. Finally, we compare our ensemble model for proper noun embedding with the models not handling proper noun as a special token and Malt parser.

4.1. Dataset

We used the dataset of "2018 Korean Language Information Competition: Development and Application of Dependency Parsing System" [32]. This dataset was automatically converted from the 21st Sejong treebank corpus for dependency parsing task. Table 2 shows the size of the dataset for train and test (i.e. the number of words and sentences). Sentences are distinguished by line-feed characters and then the dataset is composed of one or more word information by using tab character similar to the universal dependency format1. In other words, the word information contains several fields which are word index which is integer starting from one for each new sentence, head word index, dependency label, and morpheme. For proper noun embedding model, we divide test data, 5,817 sentences, into 1,045 sentences with proper nouns and 4,772 sentences without proper nouns.

4.2. Model Selection for Proper Noun Embedding

In order to choose the bidirectional LSTM model relevant for proper noun embedding, without dealing with proper noun as a special token, we experimented two compositional architectures of the bidirectional LSTM in Fig. 1 by changing embedding unit (e.g. word, syllable, and morpheme). In order to determine whether or not part-of-speech information and dropout [33] is used and the number of bidirectional LSTM layer, while varying them we experimented not using proper noun embedding. For the model training, we set the hyper-parameters with batch size of 200 as shown in Table 3. In order to evaluate the performance of Korean dependency parsing, we measured UAS (Unlabeled Attachment Score) which finds only the head location and LAS (Labeled Attachment Score) which finds the relation with head location.

Table 2. The dataset for train and test.

	The number of sentences	The number of words
Train data	53,832	602,315
Test data	5,817	57,738

Table 3. Hyper-parameters for model selection.

Hyper-parameter	Value
Embedding size	300
Hidden size	300
The number of layers	2
Optimizer	Adam
Learning rate	0.001

Fig. 3 shows the change in accuracy according to training iteration. As training progresses, the accuracy increases, and it shows the highest accuracy when the number of iterations is about 15. After that, the accuracy gradually decreased, because the parameters are overfitted to the training data. Therefore, the performance was measured when both UAS and LAS accuracy had the largest values.

To choose the model for proper noun embedding model and which multi-layer structure of bidirectional LSTM to choose as the best model among two models of Fig. 1, we compare the results by changing embedding unit (e.g. word, syllable, and morpheme), and the usage of part-of-speech information. As shown in Table 4, for the embedding unit, syllable embedding unit is better than word embedding unit. Furthermore, in order to compare syllable embedding unit with morpheme embedding unit, we implement morpheme embedding unit on the model which is best case in syllable embedding unit. As a result, we got the best performance with syllable embedding unit. The resulting performance with morpheme embedding unit is worse than that of syllable embedding unit, however, better than that of word embedding unit. Also, it shows marginal variation between syllable and morpheme embedding unit.

Table 4 also shows the performance variation according to whether the part of speech is used or not. From the Table

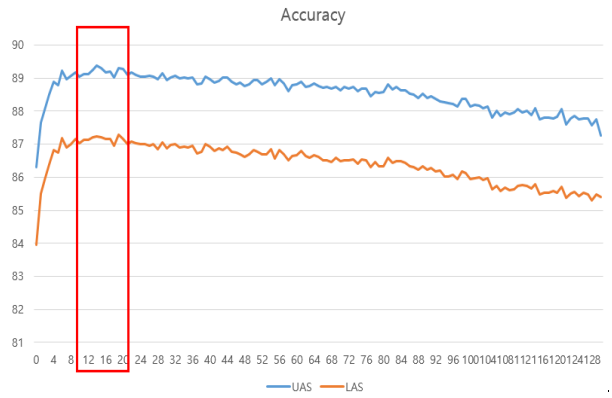

Fig. 3. Variation curve by training iteration.

[1] https://universaldependencies.org/format.html

Table 4. Accuracy not dealing with the proper noun as a special token.

Embedding unit	Part-of-speech	Model ARCHITECTURE	UAS	LAS
Syllable	Yes	(a)	89.14	86.98
		(b)	89.62	87.68
	No	(a)	86.33	81.93
		(b)	86.84	82.38
Word	Yes	(a)	87.60	85.13
		(b)	87.68	85.12
	No	(a)	79.50	68.60
		(b)	80.40	69.85
Morpheme	Yes	(b)	89.45	87.38

4, using the part-of-speech information has a tendency to enhance the performance of both UAS and LAS irrespective of whether the embedding unit is syllable or word. As for choosing the appropriate multi-layer structure of bidirectional LSTM between two compositional architectures, Fig. 1(b) shows better performance rather than that of Fig. 1(a). In Table 4, we choose the usage of part of speech information as 'yes' and the composition al architecture of bidirectional LSTM as Fig. 1(b). For the embedding unit, since the variation of the performance between syllable and morpheme embedding unit is slight, we choose both of them.

In order to choose the usage of dropout and the number of bidirectional LSTM layers for proper noun embedding model, varying the number of bidirectional LSTM layers and applying dropout as shown in Table 5, we experimented on the model which shows the best performance in Table 4. We set an embedding unit as syllable, part-of-speech as 'yes', and the compositional architecture of bidirectional LSTM as Fig. 1(b). As shown in Table 5, the performance of two layers is better than that of one layer or three layers. Although a regularization such as dropout technique has a tendency to improve the performance of the model, when the number of bidirectional LSTM layer is one or two, the dropout technique makes the model deteriorate. Compared with it, when the number of layers was three, we found that using the dropout technique enhances the performance. However, two layers of bidirectional LSTM are better than three layers of bidirectional LSTM regardless of the usage of dropout.

For the proper noun embedding model experiment, we built 2-layered bidirectional LSTM. Furthermore, we investigated the errors according to the sentence length, part-of-speech information, and long dependency errors. For 5,817 sentences, the shortest sentence has two words and the length of the longest sentence is 50. When the sentences are classified by dividing the section based on 5 words, the sentences with 16 to 20 words and the sentences with 20 or more were similar in number. So, 20 or more sentences were merged into the same section. In the case of errors according to the length of the sentence, a relatively small error rate of about 10% was shown in five words but the error rate increased as the length increased. Longer sentence has the possibility to have an error. Analyzing the sentences with more than 20 words for 3,099 sentences, the accuracy in sentence level was 53.27%.

We also investigated the types of errors that were incorrectly analyzed in relation to parts of speech. With 2,718 errors, we analyzed the part-of-speech between governor and dependent that incorrectly detected dependency relation. Analyzing the dependency errors, we had difficulty in finding three governors: "verb + cordinative_ending" (VV|EC), "verb + genitive_ending" (VV|ETM), and "common noun" (NNG). If we compare the actual governor with the predicted governor, we can see that they are generally the same part of speech. It is because the relation between words on the short distance is firstly considered rather than that between words on the long distance.

4.3. Proper Noun Embedding Model

For proper noun embedding model according to Table 4 and Table 5, we set the number of bidirectional LSTM layer as two, the compositional architecture of bidirectional LSTM as Fig. 1(b), and the use of the part of speech information as 'Yes'. For hyper-parameters of proper noun embedding model, we set a batch size of 500.

We measured UAS and LAS as the metrics of proper noun embedding model. Varying preprocessing and embedding unit (e.g. syllable or morpheme embedding unit), Table 6 shows the performance of proper noun embedding model with the sentences containing proper noun (i.e. 1,045 sentences with proper nouns of 5,817 sentences in test sentences). The preprocessing in Table 6 denotes whether proper nouns is replaced with a special token or not. For example, if the preprocessing is true, it denotes that proper nouns in both train and test corpus are replaced with a special token to train and evaluate our model. In other words,

Table 5. Accuracy with bidirectional LSTM layers and dropout.

The number of layers	Dropout	UAS	LAS
1	1.0	89.21	87.12
	0.7	89.13	86.68
2	1.0	89.62	87.68
	0.7	89.48	87.38
3	1.0	89.17	86.99
	0.7	89.38	87.24

Table 6. Accuracy with sentences including proper noun.

Preprocessing	Embedding unit	UAS	LAS
True	Syllable	87.74	85.96
	Morpheme	87.42	85.70
False	Syllable	87.44	85.83
	Morpheme	87.15	85.35

replacing the proper nouns with a special token depends on whether the preprocessing is true or not.

From the result in Table 6, for the test sentences including proper nouns, the performance of models which replace proper noun with a special token is better than that of models that do not substitute proper noun with a special token in both syllable and morpheme embedding unit. In contrast to Table 6, to evaluate the proper noun embedding model on sentences that do not contain proper nouns, we experiment the proper noun embedding model in test corpus that does not include proper nouns (i.e. 4,772 sentences of 5,817 sentences in test sentences) varying the preprocessing condition. Table 7 shows the performance of models according to preprocessing of proper noun with the test corpus that does not have proper noun, the meaning of preprocessing and embedding unit is the same from Table 6, The difference from Table 6, is only that test sentences don't include proper nouns. The results of Table 7 show that proper noun embedding model that do not replace proper noun with a special token is better than that without substitution.

When evaluating sentences in test corpus with proper noun, we confirm that after substituting proper noun with a special token in train and test corpus, syllable embedding model outperforms the model based on morpheme embedding under the same preprocessing of proper noun embedding model. Besides, when evaluating sentences in test corpus that does not contain proper nouns, we found that it is not necessary to replace a proper noun with a special token in train and test corpus. Furthermore, the results of Table 7 show that the morpheme embedding model is better than syllable embedding model on sentences that do not have proper nouns. In other words, those results imply that syllable embedding is better than morpheme embedding in handling sentences with proper noun, but morpheme embedding is better than syllable embedding in handling sentence without proper noun. So, we construct an ensemble model by integrating both of embedding models in accordance with whether sentences contain proper nouns or not. Specifically, we use syllable embedding model in sentences with proper nouns by substituting the proper nouns with a special token, and morpheme embedding model, which does not preprocess proper nouns, in sentences that do not include proper nouns.

Table 7. Accuracy with sentences not including proper noun.

Preprocessing	Embedding unit	UAS	LAS
True	Syllable	90.27	88.10
	Morpheme	90.58	88.44
False	Syllable	90.40	88.35
	Morpheme	90.72	88.58

Table 8. The comparison of performance.

Model	UAS	LAS
Word embedding model	87.68	85.12
Syllable embedding model	89.62	87.68
Morpheme embedding model	89.45	87.38
Ensemble model(our proposed model)	<u>89.93</u>	<u>87.88</u>
Malt parser: arc-eager	88.24	85.71
Malt parser: arc-standard	88.15	85.58
Lee et al.[21]	89.56	87.35

Table 8 shows the comparison of our ensemble model with models (i.e. word, syllable, and morpheme embedding models) which do not take into account the preprocessing of proper nouns based on the model with part-of-speech information and the compositional architecture of bidirectional LSTM such as Fig. 1(b). We also use as baseline transition-based Malt parser which showed excellent performance in CoNLL shared task [34]. To compare our ensemble with the baseline, Malt parser, it is implemented with both the arc-eager and the arc-standard approach. As shown in Table 8, for malt parser, the arc-eager approach surpasses the arc-standard approach. The Malt parser, irrespective of whether transition-based method is arc-eager or arc-standard, even outperforms word embedding model based on neural network in both LAS and UAS. Both syllable embedding and morpheme embedding model are better than malt parser. Furthermore, the ensemble model shows better performance than models that do not handle a proper noun as a special token that used ReLU and dropout in both UAS and LAS.

V. CONCLUSION

We proposed proper noun embedding model in the Korean dependency parsing by constructing multi-layer bidirectional LSTM and pre-training proper nouns as special tokens for an arc-eager transition-based approach. We conducted an experiments for the sentences containing proper

nouns and the sentences not containing proper nouns. As a result, the syllable-based neural network with proper noun preprocessing shows better performance than the morpheme-based one in handling the sentences with proper nouns, and morpheme embedding-based neural network without proper noun preprocessing shows better performance than the syllable-based one in handling the sentences not including the proper nouns. When we compared with Malt parser, arc-eager approach is better than arc-standard approach in Malt parser. Comparing the Malt parser with the models not handling proper nouns as a special token, Malt parser outperforms word embedding model based on neural network, but deteriorate than both syllable embedding model and morpheme embedding model. Finally, our ensemble model, integrating syllable embedding and morpheme embedding according to existence of proper noun, shows better performance than Malt parser. The result shows that the performance improves by 1.69%p for UAS and 2.17%p for LAS than Malt parser's arc-eager approach using the same transition-based method in proper noun embedding model.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (NRF-2021R1F1A1061433). This paper is an extended from the conference paper [31] with further experiments in the first author's MS thesis [35].

REFERENCES

[1] J. Nivre, "Algorithms for deterministic incremental dependency parsing," *Computational Linguistics*, vol. 34, no. 4, pp. 513-553, Dec. 2007.

[2] J. Nivre, "Non-projective dependency parsing in expected linear time," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore, Aug. 2009, pp. 351-359, 2009.

[3] T. Kudo and J. Richardson, "Sentence piece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussels, Belgium, Nov. 2018, pp. 66-71.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," https://arxiv.org/abs/1301.3781.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, USA, Dec. 2013, pp. 3111-3119.

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota, Jun. 2019, pp. 4171-4186.

[7] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations," https://arxiv.org/abs/1909.11942.

[8] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[10] T. Brown, et al., "Language models are few-shot learners," in *Advances in Neural Information Processing systems*, Dec. 2020, pp. 1877-1901.

[11] Z. Hu, T. Chen, K. Chang, and Y. Sun, "Few-shot representation learning for out-of-vocabulary words," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, Jul. 2019, pp. 4102-4112.

[12] T. Lee and S. Kang, "Automatic text summarization based on selective OOV copy mechanism with BERT embedding," *Journal of KIISE*, vol. 47, no. 1, pp. 36-44, Jan. 2020.

[13] D. Weiss, C. Alberti, M. Collins, and S. Petrov, "Structured training for neural network transition-based parsing," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Beijing, China, Jul. 2015, pp. 323-333.

[14] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, and K. Ganchev, et al., "Globally normalized transition-based neural networks," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, Aug. 2016. pp. 2442-2452.

[15] H. Wang, H. Zhao, and Z. Zhang, "A transition-based system for universal dependency parsing," in *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Vancouver, Canada, Aug. 2017, pp. 191-197.

[16] W. Wang and B. Chang, "Graph-based dependency parsing with bidirectional LSTM," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, Aug. 2016,

pp. 2306-2315.

[17] Z. Zhang, H. Zhao, and L. Qin, "Probabilistic graph-based dependency parsing with convolutional neural network," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, Aug. 2016. pp. 1382-1392.

[18] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, Oct. 2014, pp. 740-750.

[19] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Beijing, China, Jul. 2015, pp. 334-343.

[20] M. Ballesteros, C. Dyer, and N. A. Smith, "Improved transition-based parsing by modeling characters instead of words with LSTMs," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, Sep. 2015, pp. 349-359.

[21] C. Lee, J. Kim, and J. Kim, "Korean dependency parsing using deep learning," in *Proceedings of the 26th Annual Conference on Human and Cognitive Language Technology*, 2014, pp. 87-91.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, Nov. 1997.

[23] E. Kiperwasser and Y. Goldberg, "Simple and accurate dependency parsing using bidirectional LSTM feature representations," in *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 313-327. 2016.

[24] J. Li and J. H. Lee, "Korean transition-based dependency parsing with recurrent neural network," *KIISE Transactions on Computing Practices*, vol. 21, no. 8, pp. 567-571, 2015.

[25] S. H. Na, K. Kim, and Y. K. Kim, "Stack LSTMs for transition-based Korean dependency parsing," in *Proceedings of the Korea Computer Congress*, 2016, pp. 732-734.

[26] Z. Zhang, S. Liu, M. Li, M. Zhou, and E. Chen, "Stack-based multi-layer attention for transition-based de pendency parsing," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, Sep. 2017, pp. 1677-1682.

[27] J. W. Min,and S. H. Na, "SyntaxNet models using transition based recurrent unit for Korean dependency parsing," in *Proceedings of the Korea Computer Congress'17*, 2017, pp. 602-604.

[28] C. Park and C. Lee, "Korean dependency parsing by using pointer networks," *Journal of Korean Institute of Information Scientists and Engineers*, vol. 44, no. 8, pp. 822-831, 2017.

[29] Z. Li, J. Cai, S. He, and H. Zhao, "Seq2seq dependency parsing," in *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA, Aug. 2018, pp. 3203-3214.

[30] J. H. Lim, Y. C. Yoon, Y. J. Bae, S. J. Lim, H. K. Kim, and K. C. Lee, "Korean dependency parsing model based on transition system using head-final constraint," in *Proceedings of the 26th Annual Conference on Human and Cognitive Language Technology*, 2014, pp. 81-86.

[31] G. H. Nam, H. Lee, and S. Kang, "Korean dependency parsing with proper noun encodin," in *Proceedings of the 2019 4th International Conference on Intelligent Information Technology*, Da Nang, Viet Nam, Feb. 2019, pp. 113-117.

[32] C. Lee, J Bae, C. Park, H. Hong, and S. Lee, "Korean information processing system competition: Korean dependency parsing," in *Proceedings of the 30th Conference of Human and Language Technology*, 2018, pp. 675-677.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929-1958, 2014.

[34] J. Nivre, J. Hall, and J. Nilsson, "MaltParser: A data-driven parser-generator for dependency parsing," in *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, Genoa, Italy, May 2016, pp. 2216-2219.

[35] G. H. Nam, "Transition-based deep learning approach to Korean dependency parsing," M.S. thesis, Kookmin University, Seoul, Korea, 2018.

AUTHORS

Gyu-Hyeon Nam received his B.S. and M.S. degrees in the Department of Computer Engineering from Kookmin University, Korea, in 2017 and 2019, respectively. Since 2019, He has been working at DeepBrain AI, Korea. His research interests include speech synthesis and natural language processing.

Hyun-Young Lee received his B.S. degree and M.E. degree in Computer Science from Kookmin University in 2016 and 2019 respectively. He received Ph.D. degree in Computer Science from Kookmin University in 2022. Currently, he is working for KT Corporation. His research interests include natural language processing, machine learning, deep learning, recommendation system, and information retrieval.

Seung-Shik Kang received his B.S. degree in Computer Science from Seoul National University in 1986, and M.S. and PhD degrees in Computer Science from the same University, in 1988 and 1993, respectively. He worked for Hansung university as an associate professor from 1994 to 2001. Currently, he is working for Kookmin University as a full professor. His research interests include natural language processing, information retrieval, text mining, big data processing, and machine learning.