International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 12 (2019), pp. 2775-2779
© International Research Publication House. http://www.irphouse.com

Deep Learning-based End-to-End Dependency Parsing without POS Tagging

Ji-un Jeon, Do-Heon Choi and So-Young Park

*Game Design and Development, Sangmyung University,
Seoul, Republic of Korea.*

*ORCID: 0000-0002-3049-9549 (Ji-Un Jeon)
ORCID: 0000-0001-5128-9981 (Do-Heon Choi)
ORCID: 0000-0003-0746-218X (So-Young Park)*

Yeo-Chan Yoon

*SW & Contents Research Laboratory,
ETRI, Daejeon. Republic of Korea.*

ORCID 0000-0002-5573-8964

Abstract

In this paper, we propose a deep learning-based end-to-end dependency parsing model using syllable-based word representation. Part-of-speech tagging additionally requires resources, processing time, and memory, while the part-of-speech tagging errors can be propagated into the dependency parsing errors. The proposed syllable-based word representation method excludes the separate part-of-speech tagger from dependency parsing. For faster parsing, the proposed model allows only a single candidate, rather than all possible candidates. Considering the long-distance dependency between a head and its dependent, the proposed model adopts the stack-pointer network, one of the state-of-the-art deep learning methods. Experimental results show that the unlabeled attachment score of the proposed parsing model without the separate part-of-speech tagger is 88.64%, which is comparable to 89.04% of the parsing model with the part-of-speech tagger. Moreover, the proposed model requires 80 milliseconds per sentence, faster than 140 milliseconds of the parsing model with the part-of-speech tagger. Besides, the proposed model requires 3,319 MB in memory, less than 4,840 MB of the parsing model with the part-of-speech tagger.

Keywords: Deep Learning, Dependency Parsing, End-to-End Learning, Natural Language Processing, Word Representation

I. INTRODUCTION

As smart applications, such as text mining and AI speakers, rapidly develop in the recent years, they are asked to provide the desired services by understanding the meaning of a sentence represented by a user [1,2]. Therefore, parsing a sentence is very important. The parser analyzes the dependency between a head word and its dependent in the sentence after identifying every word in that sentence. However, it is difficult to parse the sentence, because the sentence can include some ambiguities. For example, the word “saw” has three different meanings: the past tense of the transitive verb “see” (to look), the present tense of the intransitive verb “saw” (to cut wood or metal), and the noun “saw” (a tool used for cutting wood or metal). In the sentence “Rabbit saw the turtle walked so slowly,” the head words of “the turtle” and “slowly” can be either “saw” or “walked.”

To reduce the complexity of parsing the sentence, most of previous parsing approaches have been divided into part-of-speech tagging to assign a part-of-speech tag to every word in the sentence and dependency parsing to analyze every dependency between the head word and its dependent in the sentence [3-9]. Although the part-of-speech tagging is very useful for dependency parsing, it also has the following weaknesses for the dependency parsing. First, it needs resources, such as a dictionary. Second, it additionally requires processing time and memory. Third, the part-of-speech tagging errors can be propagated into the dependency parsing errors.

In this paper, we propose an end-to-end dependency parsing model without the separate part-of-speech tagger. Section 2 describes some previous dependency parsing approaches. Section 3 introduces the stack-pointer network-based dependency parsing model. Section 4 explains the proposed syllable-based word representation method, substituted for the separate part-of-speech tagger. Section 5 shows the experimental results of the proposed dependency parsing model. Section 6 concludes the paper.

II. RELATED WORKS

Dependency parsing approaches have recently evolved from statistical approaches into deep learning-based approaches using the word embedding to alleviate the data sparseness problem [3-9]. They are classified as follows. First, graph-based dependency parsing approaches generate all possible candidates from a sentence before they finally select the most possible candidate by calculating the probability of generating every candidate based on the statistical information [3]. To alleviate the data sparseness problem, the graph-based parsing approaches using the deep learning method utilize the word embedding, instead of a one-hot vector [3,4]. However, they tend to be slow because they can generate too many candidates. Moreover, the part-of-speech tagging errors are propagated into the dependency parsing errors; because they depend on the part-of-speech tagging.

Second, transition-based dependency parsing approaches generate a single candidate from the sentence because they select the shift–reduce transition action based on the buffer and the stack [5,6]. Unlike the graph-based approaches, they do not allow all possible candidates per ambiguous state; hence, they quickly parse a sentence in a deterministic linear time. Like the graph-based approaches, they use the separate part-of-speech tagger. To calculate the score of selecting a transition action between a shift and a reduce, the statistical models use the statistical information, while the deep learning-based models use the deep neural network. For improving the long-distance dependency problem, they utilize the long short-term memory (LSTM) [6].

Third, stack-pointer network-based dependency parsing approaches generate a candidate from the sentence by selecting a stack action between the push and pop actions per ambiguous state [7,8]. They calculate the score by using the pointer network method to learn the conditional probability [7,9]. Like the transition-based approaches, they select the most plausible candidate per ambiguous state; hence, they parse a sentence in deterministic linear time. Also, they depend on the part-of-speech tagger. To improve the long-distance dependency problem, the stack-pointer network-based dependency parsing approaches can utilize the information of every word in the sentence per ambiguous state.

In this paper, we propose a stack-pointer network-based end-to-end dependency parsing model without the separate part-of-speech tagger. Considering the processing time, we adopt the transition-based approach to allow only a single candidate, rather than the graph-based approach to allow all possible candidates. Moreover, we choose the stack-pointer network-based approach to improve the long-distance dependency problem. Unlike the previous approaches, the proposed end-to-end parsing model does not depend on the separate part-of-speech tagger.

III. STACK-POINTER NETWORK

The stack-pointer network-based dependency parsing model [7,8] consists of an encoder, a decoder, and a parser, as showed in Fig. 1. The encoder assigns more context information to each word in a sentence, while the decoder interprets the encoded word information to find the dependent of the head word. Given a head word on the current top of the stack, the parser pushes the most plausible dependent candidate onto the new top of the stack by analyzing the decoded information representing the relation between the head word and its each dependent candidate. When the parser finds no dependent of the head word, the parser pops the head word from the top of the stack. Finally, the parser generates a dependency parse tree with the stack actions, such as push and pop.

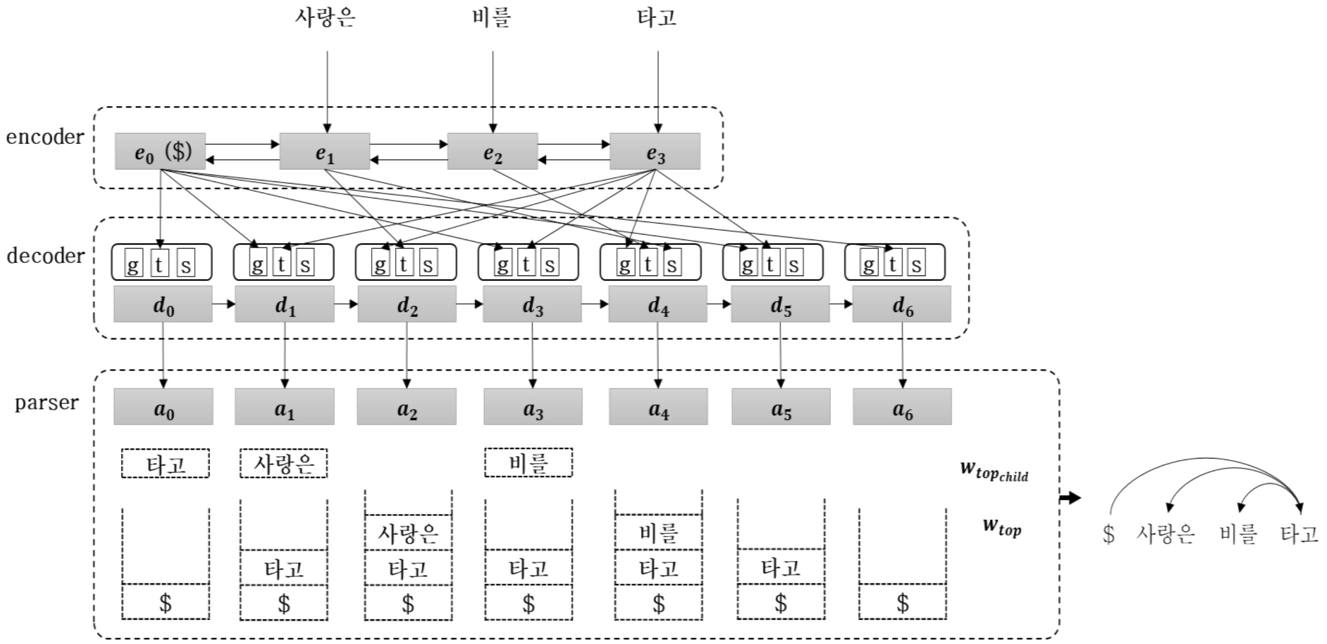

Figure 1. Stack-pointer network

$$Tree = \underset{Tree}{argmax}\ Score(\,Tree \mid w_{1n}\,) \overset{\text{def}}{=} \underset{e_{0,\mathrm{n}},d_{0,2\mathrm{n}},a_{0,2\mathrm{n}}}{argmax}\ Score(\,e_{0,\mathrm{n}},\, d_{0,2\mathrm{n}}, a_{0,2\mathrm{n}} \mid w_{1n}\,) \qquad (1)$$

$$\begin{aligned} &Score(\,e_{0,\mathrm{n}},\, d_{0,2\mathrm{n}}, a_{0,2\mathrm{n}} \mid w_{1n}\,) \\ &\quad = Score(\,e_{0,\mathrm{n}} \mid w_{1n}) \times Score(\,d_{0,2\mathrm{n}} \mid e_{0,\mathrm{n}}, w_{1n}) \times Score(\,a_{0,2\mathrm{n}} \mid d_{0,2\mathrm{n}}, e_{0,\mathrm{n}}, w_{1n}\,) \\ &\quad \approx \prod_{i=0}^{n} Score(\,e_{\mathrm{i}} \mid w_{1n}) \times \prod_{i=0}^{2n} Score(\,d_i \mid e_{0,\mathrm{n}}, w_{1n}) \times \prod_{i=0}^{2n} Score(\,a_i \mid d_i, e_{0,\mathrm{n}}, w_{1n}) \qquad (2) \end{aligned}$$

As described in Eq. (1), the proposed model generates the dependency parse tree $Tree$ with the highest score from the words w_{1n} in the sentence. It consists of an encoder, a decoder, and a parser; hence, the generated dependency parse tree $Tree$ can be redefined as the sequence of the encoded word representations $e_{0,n}$, the decoded information $d_{0,2n}$, and the stack actions $a_{0,2n}$. As a result of both pushing and popping the words w_{1n} in the sentence, the number of selecting the stack actions indicates $2n$ representing two times the number of all the words in the sentence. Eq. (2) assumes that the score function can generalize multiple events by the chain rule, and some independent information exists.

$$\begin{aligned} e_i = \underset{e_i}{argmax}\ Score(e_i|w_{1n}) \ &\approx\ \vec{e}_i \circ \overleftarrow{e}_i \\ &= \ LSTM_{fore}(w_i, \vec{e}_{i-1}, \vec{c}_{i-1}) \\ &\circ LSTM_{back}(w_i, \overleftarrow{e}_{i+1}, \overleftarrow{c}_{i+1}) \qquad (3) \end{aligned}$$

Given an ambiguous word, the encoder allows only a single encoded candidate e_i for the i-th word w_i, as represented in Eq. (3). Moreover, the encoder with a bidirectional LSTM model concatenates both the output $\vec{e}_i$ of the forward LSTM from the first word to the i-th word and the output $\overleftarrow{e}_i$ of the backward LSTM from the last word to the i-th word. The encoder can consider the context of the word, such as the previous word w_{i-1} and the next word w_{i+1}, because it utilizes the encoded word information $\vec{e}_{i-1}$ and $\overleftarrow{e}_{i+1}$ and the cell memory $\vec{c}_{i-1}$ and $\overleftarrow{c}_{i+1}$.

$$\begin{aligned} d_i \ = \ \underset{d_i}{argmax}\ Score(\, d_i \,|\, e_{1n}, w_{1n}) \\ \approx \ LSTM_{decoder}\Big(e_{top} + e_{top_{grand}} \\ + e_{top_{sibilng}}, d_{i-1}, c_{i-1}\Big) \qquad (4) \end{aligned}$$

As represented in Eq. (4), the decoder also allows a single decoded candidate d_i for the i-th stack action. Because the decoder adopts a single-directional LSTM model, it can utilize the previously generated results such as the encoded word information, and the decoded information. Eq. (4) describes that it yields the decoded candidate d_i using the encoded word information e_{top} of the word w_{top} on the current top of the stack, the encoded word information $e_{top_{grand}}$ of its grandparent, the encoded word information $e_{top_{sibilng}}$ of its siblings, the previous decoded information d_{i-1}, and the previous cell memory c_{i-1}.

$$\begin{aligned} a_i \ = \ \underset{a_i}{argmax}\ Score(\, a_i \,|\, d_i,\, e_{1n}, w_{1n}\,) \\ = \underset{w_{top_{child}}}{argmax}\ Score\big(\, w_{top_{child}} \,\big|\, d_i, e_{1n}, w_{1n}\,\big) \\ = \begin{cases} push\big(w_{top_{child}}\big) & if\ w_{top_{child}}\ exists \\ pop(\,) & otherwise \end{cases} \qquad (5) \end{aligned}$$

The parser selects the most plausible stack action a_i, such as push ($w_{top_{child}}$) or pop ( ), after calculating the action score based on the encoded word information $e_{0,n}$ and the decoded information d_i representing the relations between the head word w_{top} on the current top of the stack and its dependent word $w_{top_{child}}$, as described in Eq. (5). When the most plausible dependent word $w_{top_{child}}$ exists in the sentence, the parser pushes the dependent word onto the new top of the stack. Otherwise, the parser pops the head word w_{top} from the current top of the stack.

$$\begin{aligned} w_{top_{child}} &= Attention(\, d_i, e_{1n}) \approx MHDPA(Q, K, V) \\ &= softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V \qquad (6) \end{aligned}$$

In order to choose the most appropriate dependent word $w_{top_{child}}$ of the head word w_{top} from all words in the sentence, the parser adopts the MHDPA (multi head dot product attention), one of the most recent attention methods [10], to obtain a probability distribution. For a more precise estimation, the MHDPA method performs the scaled dot-product attention after allowing multiple heads. In Eq. (6), Q, K, V, and d_K denote queries, keys, values, and the dimensionality of the key vectors used as a scaling factor, respectively. The decoded information d_i corresponds to the queries Q, while the concatenation results $e_1 \circ e_2 \ldots \circ e_n$ of the encoded information e_{1n} correspond to the keys (K) and the values (V).

IV. SYLLABLE-BASED WORD REPRESENTATION WITHOUT POS TAG

The proposed end-to-end dependency parsing model represents the i-th word w_i in the sentence as the combinations of its syllables $s_1, s_2, \ldots, s_{l-1}, s_l$, as described in Fig. 2 and Eq. (7). After calculating the dot product between the word vector and the weight matrix W_0, and between the syllable vectors and the weight matrix W_s, it integrates the calculated vectors with the integration weight matrix W_I, similar to previous word representation methods[6,8,9]. To improve the unknown word problem, the proposed model utilizes CNN mapping from the high-dimensional vector to the low-dimensional vector, after concatenating all syllable vectors to form the single high-dimensional vector.

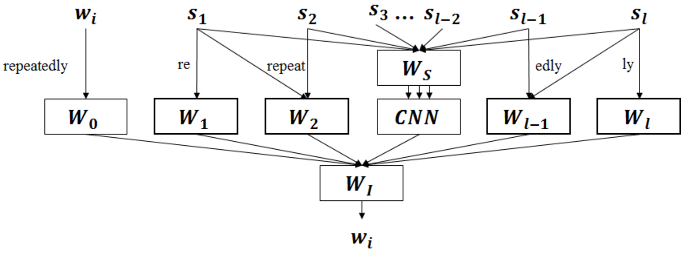

Figure 2. Syllable-based word representation method

Table-1: End-to-End Parsing Performance (%) According to Syllable-based Word Representation

	base line	W_1 W_2	W_1 W_{l-1}	W_1 W_l	W_2 W_{l-1}	W_2 W_l	W_{l-1} W_l	W_1 W_2 W_{l-1}	W_1 W_2 W_l	W_1 W_{l-1} W_l	W_2 W_{l-1} W_l	W_1 W_2 W_{l-1} W_l
UAS	76.4	80.0	85.3	85.8	85.0	86.0	86.5	85.8	87.0	87.8	87.8	88.6
LAS	57.5	64.4	77.6	79.4	77.2	79.7	80.9	78.5	81.3	82.9	82.9	84.2

Unlike the previous method [3-9] calculating the dot product between the part-of-speech tag vector and the weight matrix, the proposed end-to-end dependency parsing model excludes the separate part-of-speech tagger; because the part-of-speech tagger additionally requires resources, processing time, and memory, while the part-of-speech tagging errors can be propagated into the dependency parsing errors. Instead of the separate part-of-speech tagger, the proposed model utilizes the syllable-based word information: the dot products between the first syllable vector and the weight matrix W_1, between the first two syllable vectors and the weight matrix W_2, between the last two syllable vectors and the weight matrix W_{l-1}, and between the last syllable vector and the weight matrix W_l.

$$w_i' = W_I\Big((W_0 \cdot w_i) \circ CNN(W_s \cdot s_1, \dots, W_s \cdot s_l) \circ (W_1 \cdot s_1) \circ (W_2 \cdot s_{1,2}) \circ (W_{l-1} \cdot s_{l-1,l}) \circ (W_l \cdot s_l)\Big) \quad (7)$$

As shown in Fig. 2, for example, the English word w_i “repeatedly” consists of four syllables: the first syllable s_1 “re-” meaning “to do something again”, the second syllable s_2 “peat” derived from the Latin “petere” meaning “to seek”, the second last syllable s_{l-1} “-ed” used to create the past tense form, and the last syllable s_l “-ly” used to form adverbs. A word consists of some morphemes, and each morpheme serves one purpose in agglutinative languages, such as in the Korean language [6,8,9], in which too massive words exist. Considering that the meaning or the function of the word can be inferred from few syllables, the proposed model particularly focuses on the first two syllables and the last two syllables in the word.

V. EXPERIMENTS

To analyze the performance of the proposed end-to-end dependency parsing model without the separate part-of-speech tagger, we implement some models based on the stack-pointer network [7,8], as presented in Table 1 and Table 2. All embedding matrixes were initialized according to Bernoulli distribution with word frequency. Also, we divide Sejong corpus with 59,659 sentences into a training set with 53,842 sentences (90%) and a test set with 5,817 sentences (10%) [6,8,9]. The unlabeled attachment score (UAS) is the ratio of the correct candidates from all the unlabeled arcs in the test corpus, while the labeled attachment score (LAS) is the ratio of the correct candidates from all the labeled arcs in the test corpus. In the measures, the candidate indicates the arc generated by the model, and the number of labels is 39 in the corpus [6,8,9]. The LAS checks both the label and the head word per arc, while the UAS checks the head word.

Table 1 shows the performances of the end-to-end parsing model without the separate part-of-speech tagger according to the syllable combinations. The baseline model performed at 76.4% on the UAS and 57.5% on the LAS, while the proposed best model performed at 88.64% on the UAS and 84.27% on the LAS. As compared with the baseline model, the proposed best model improved by 12.2% on the UAS and 26.7% on the LAS. The result describes that the baseline with the embedding matrixes W_I, W_S, and W_0 can improve the parsing performance by using the syllable-based embedding matrixes W_1, W_2, W_{l-1}, and W_l.

Table-2: Parsing Performance with or without Part-of-speech Tagging

Model	UAS	LAS	Parsing Time (sec)	Time per Sentence (sec)	Memory (MB)
Parsing without POS tag	76.40%	57.47%	509.30	0.087	3,240
Parsing with POS tagger	89.04%	86.13%	845.91	0.145	4,840
Proposed End-to End Parsing	88.64%	84.27%	**523.46**	**0.089**	**3,319**

Table 2 shows that the parsing model with the KKMA part-of-speech tagger takes 89.04% on the UAS and 86.13% on the LAS much better than the performance of the baseline model without the part-of-speech tagger. It describes that the part-of-speech tagging is very useful for the dependency parsing. In Table 2, the parsing time indicates the time of parsing 5,817 sentences in the test corpus on ubuntu system with Intel I7-7700 3.60GHz CPU, 2 GTX 2080, and RAM 32GB. Since the proposed end-to-end parsing model does not need the separate part-of-speech tagger, it requires 80 milliseconds per sentence, faster than 140 milliseconds of the parsing model with the part-of-speech tagger. Besides, it requires 3,319 MB, less than 4,840 MB of the parsing model with the part-of-speech tagger. Nevertheless, the proposed end-to-end parsing model performs at 88.64% on the UAS, which is comparable to 89.04% of the parsing model with the separate part-of-speech tagger.

VI. CONCLUSION

In this paper, we propose a deep learning-based end-to-end dependency parsing model using the syllable-based word representation. The proposed model has the following characteristics. First, the proposed model is inexpensive in construction cost; because it is applied to the end-to-end learning architecture, in which both the syllable-based word representation step and the stack-pointer network-based parsing step can learn from the corpus all at once.

Second, the proposed model is efficient in terms of the processing time and memory; because it excludes the separate part-of-speech tagger, and allows only a single candidate, rather than all possible candidates. The experimental results showed that the proposed model requires 80 milliseconds per sentence, faster than 140 milliseconds of the parsing model with the separate part-of-speech tagger. Besides, it requires 3,319 MB, less than 4,840 MB of the parsing model with the part-of-speech tagger.

Third, the proposed model roughly correctly parses the sentence; because it adopts the stack-pointer network, one of the state-of-the-art deep learning methods, and the proposed syllable-based word representation method is effectively substituted for the separate part-of-speech tagger. The experimental results showed that the proposed model without the part-of-speech tagger achieved 88.64% on the UAS, which is less than 89.04% of the parsing model with the part-of-speech tagger.

For the future work, we will apply the proposed deep learning-based end-to-end dependency parsing model to smart agent systems, such as an AI speaker. We will then receive and overcome the unexpected actual parsing problems in real applications. We will also study how to become more efficient and more correct in parsing.

ACKNOWLEDGEMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2018R1D1A1B07048122)

REFERENCES

[1] K. Batsuren, E. Batbaatar, T. Munkhdalai, M. Li, O. Namsrai and K. H. Ryu, A Dependency Graph-Based Keyphrase Extraction Method Using Anti-patterns, Journal of Information Processing Systems, 14(5), 2018, 1254-1271.

[2] Yeh, Jui-Feng, Speech act identification using semantic dependency graphs with probabilistic context-free grammars, ACM Transactions on Asian and Low-Resource Language Information Processing, 15(1), 2016.

[3] Si, N., Wang, H., & Shan, Y., Exploring global sentence representation for graph-based dependency parsing using BLSTM-SCNN, Pattern Recognition Letters, Elsevier, 105, 2018, 96-104

[4] He, R., Wang, Y., Song, D., Zhang, P., Jia, Y., & Li, A. A ,Dependency Parser for Spontaneous Chinese Spoken Language, ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 17(4), 2018, 28.

[5] Ouchi, H., Duh, K., Shindo, H., Matsumoto, Y., Ouchi, H., Duh, K., and Matsumoto, Y. Transition-based dependency parsing exploiting supertags, IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 24(11), 2016, 2059-2068.

[6] Na, S. H., Li, J., Shin, J. H., & Kim, K, Transition-Based Korean Dependency Parsing Using Hybrid Word Representations of Syllables and Morphemes with LSTMs, ACM Transactions on Asian and Low-Resource Language Information Processing, 18(2), 2018.

[7] Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. Stack-pointer networks for dependency parsing, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 1, 2018, 1403-1414.

[8] Ahn, H., Seo, M., Park, C., Kim, J., & Seo, J, Extensive Use of Morpheme Features in Korean Dependency Parsing, 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), 2019, 1-4.

[9] Jung, S., Park, C. E., & Lee, C, Multitask Pointer Network for Korean Dependency Parsing, ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 18(3) 2019, 24.

[10] Dong, L., Xu, S., & Xu, B., Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition, 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, 5884-5888.