Syntax-aware Transformer Encoder for Neural Machine Translation

Sufeng Duan[1], Hai Zhao[1,*], Junru Zhou[1] and Rui Wang[2]
[1] *Department of Computer Science and Engineering,*
Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering,
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, China
[2] *National Institute of Information and Communications Technology (NICT), Kyoto, Japan*
1140339019dsf@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn, zhoujunru@sjtu.edu.cn, wangrui@nict.go.jp

***Abstract*—Syntax has been shown a helpful clue in various natural language processing tasks including previous statistical machine translation and recurrent neural network based machine translation. However, since the state-of-the-art neural machine translation (NMT) has to be built on the Transformer based encoder, few attempts are found on such a syntax enhancement. Thus in this paper, we explore effective ways to introduce syntax into Transformer for better machine translation. We empirically compare two ways, positional encoding and input embedding, to exploit syntactic clues from dependency tree over source sentence. Our proposed methods have a merit keeping the architecture of Transformer unchanged, thus the efficiency of Transformer can be kept. The experimental results on IWSLT' 14 German-to-English and WMT14 English-to-German show that our method can yield advanced results over strong Transformer baselines.**

***Keywords*-Neural Machine Translation, dependency parsing, POS Tagging**
[1]

I. Introduction

Neural machine translation (NMT) models are based on a sequence-to-sequence (seq2seq) architecture, which uses an encoder to create a vector of source sequence and a decoder to predict target sequence, usually with an attention mechanism [1, 2, 3]. Recently, a series of seq2seq NMT models, such as recurrent neural network (RNN) model [2, 3, 4, 5], convolutional neural network [6] model and the Transformer which is empowered by self-attention mechanism [7], have been proposed and get good performance.

Seq2seq models have to take source input as sequence which makes the tree-based syntactic information difficult to add to the model directly. Inspired by incorporating syntactic information in statistical machine translation (SMT) [8, 9], syntactic information has been incorporated into NMT model for better performance [10, 11, 12, 13] in the similar ways of SMT, in which two types of approaches have been proposed, one is to still introduce tree structure directly [10], like tree long-short term memory (Tree-LSTM) and tree-attention, and the other turns to linearization of syntactic tree [11, 12, 13].

However, existing works using syntactic information are almost all for RNN-based NMT. For Transformer, its self-attention mechanism has been capable of capturing general relationship among words which is supposed to be delivered by syntax from a view of linguistics. Thus using a syntax integration method like [12], it is quite possible that no significant performance improvement from the syntactic clues will be observed. For those models introducing syntax in terms of tree structure, model architecture has to be modified and more parameters will be introduced, which brings the drawback of slowing down computational efficiency. At last, the Transformer encoder adopts scaled dot-product attention to finalize representation, which is quite different from what RNN-based encoder does, thus it is not so convenient to modify the Transformer architecture to accommodate syntactic information as the same integration way in RNN-based NMT.

In this work, we propose using syntax to enhance the Transformer based NMT in terms of input embedding and positional encoding. The first method which is inspired by the work [12] using independent syntactic embedding to enhance RNN-based NMT views the tree structure as one label and encodes it by embedding. The second method transfers tree features into positional encoding, and may conveniently keep the model architecture unchanged. Our experiments on German-to-English (De-En) and English-to-German (En-De) translation show that our proposed syntax-enhanced Transformer indeed obtains better translation performance.

II. Background

A. Syntactic Information for NMT

There are two ways to incorporate syntactic information into NMT model. The first is to use tree structure such as Tree-LSTM to directly encode syntactic information. Eriguchi et al. [10] propose a tree-to-sequence model with a tree-based encoder, which encodes the phrase structure of a sentence as vectors.

The second is to represent syntax through linearization. Linearized syntactic tree is advantageous because it is

[1]* Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100) and Key Projects of National Natural Science Foundation of China (No. U1836222 and No. 61733011).

easily added into models with minor modifications of the model architecture. Aharoni and Goldberg [11] design a sequence-to-tree model which translates source sentence to a linearized constituency tree. Sennrich and Haddow [14] propose a feature method which will be followed by this work,

$$\overrightarrow{h}_j = \tanh(\overrightarrow{W}(\big\Vert_{k=1}^{|F|} E_k x_{jk}) + \overrightarrow{U}\overrightarrow{h}_{j-1}), \tag{1}$$

where m is the word embedding size, $\overrightarrow{U} \in \mathbb{R}^{n \times n}$ are weight matrices, n is the number of hidden units, $|F|$ is the number of additional features, $\Vert$ is the operator of concatenation, $E_k \in \mathbb{R}^{m_k \times K_k}$ are the feature embedding matrices, with $\sum_{k=1}^{|F|} m_k = m$, $\overrightarrow{h}_{j-1}$ is the hidden states of the $(j-1)$-th words and K_k is the vocabulary size of the k-th feature.

B. Transformer

Transformer is introduced by [7] which is based on self-attention network and good at getting the relationship between words in sentence. The encoder of Transformer is composed of a stack of N layers, which has two sub-layers in one layer. The first sub-layer is a multi-head self-attention structure to generate the attention of input. Transformer computes the matrix of outputs as

$$\text{Attention}(H) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

where Q is the matrix of queries, K is the matrix of keys and V is the matrix of values. Queries, keys and values are transferred from the input H. Q, K and V are all generated from input representations. The second sub-layer is a position-wise fully connected feed-forward network. The Transformer employs a residual connection [15] which is followed by layer normalization [16]. The output of sub-layers in encoder is

$$\begin{aligned} \mathbf{C}^k &= \textsc{LayerNorm}\big(\textsc{Attention}(\mathbf{H}^{k-1}) + \mathbf{H}^{k-1}\big) \\ \mathbf{H}^k &= \textsc{LayerNorm}\big(\text{FFN}(\mathbf{C}^k) + \mathbf{C}^k\big) \end{aligned} \tag{3}$$

where C^k is the output of first sub-layer in the k-th layer, H^k is the output of second sub-layer in the k-th layer. The decoder is composed of a stack of N layers. Different from encoder, decoder layer adds another sub-layer to perform attention over the output of encoder.

Several variants of Transformer have been proposed. Shaw et al. [17] extend the self-attention mechanism to efficiently consider representations of the relative positions or distances between sequence elements. Xiao et al. [18] propose a lattice-based encoder to explore effective word or subword representation in an automatic way during training.

As few existing studies consider using syntax to enhance the state-of-the-art Transformer though syntax itself has been shown helpful for previous SMT and RNN-based NMT, we thus make an attempt to filling the gap by exploring a syntax-enhanced Transformer.

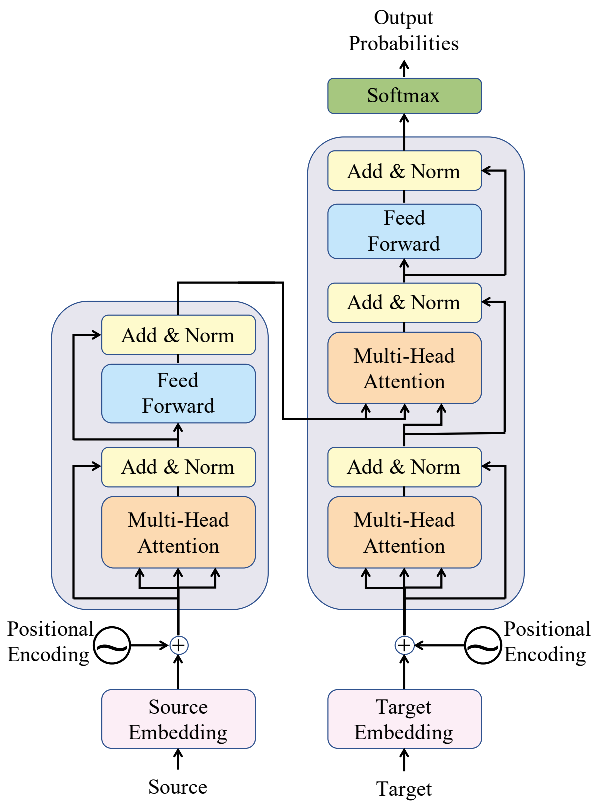

Figure 1. Architecture of Transformer

C. Dependency Parsing

Dependency parsing aims to predict the existence and type of linguistic dependency relations between words. It is a fundamental task in natural language processing (NLP) [19, 20, 21, 22, 23, 24, 25]. Dependency parsing is often used for other NLP tasks such as semantic role labeling [23, 24, 26, 27, 28]. For most east Asian languages such as Chinese, dependency parsing relies on word segmentation [29, 30, 31, 32]. It can be roughly put into two categories in terms of searching strategies over parsing trees, graph-based and transition-based [33]. With the development of neural network applied to dependency parsing, there comes continuous progress for better parsing performance [22, 34]. Zhang et al. [35] propose a neural probabilistic parsing model which explores up to third-order graph-based parsing with maximum likelihood training criteria. Li et al. [36] propose a full character-level neural dependency parser together with a released character-level dependency treebank for Chinese. The dependency parsing is shown to be more effective than non-neural parser. Wu et al. [37] propose a system for multilingual universal dependency parsing from raw text. Li et al. [38] propose a tree encoder and integrate pre-trained language model features for a better representation of partially built dependency subtrees and thus enhances the model.

III. Models

Fig. 2 shows the architecture of our model. Different from existing methods encoding syntax into the source

Table I
FEATURE EXAMPLES FOR A SENTENCE. FOR ROOT NODE, ITS DEPENDENCY LABEL IS *root*.

words	It	is	a	good	thing	for	people	.
order	1	2	3	4	5	6	7	8
POS tags	PRP	VBZ	DT	JJ	NN	IN	NNS	.
Dependency labels	sbj	root	det	amod	obj	case	nmod	punct
parent position	2	0	5	5	2	7	5	2
tree depth	2	1	3	3	2	4	3	2

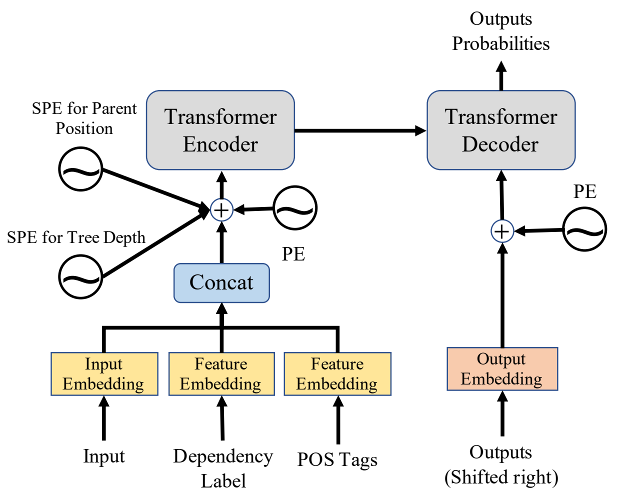

Figure 2. Architecture of our model. Our syntactic features are encoded into the input embedding or position encodings.

[10, 11], we consider two ways to encode syntactic information derived from a syntactic dependency parse tree. One is directly extracting syntactic clues as a part of input embedding, the other is regarding appropriate syntactic clues as a type of position information thus putting them into positional encodings of the Transformer. From syntactic source, we extract four types of syntactic features, including 1) POS tags, 2) dependency labels, 3) parent position and 4) tree depth of a node. The features 1-3 can be integrated into input embedding, and the features 3 and 4 can be encoded into positional encodings of the Transformer.

A. Syntactic Features

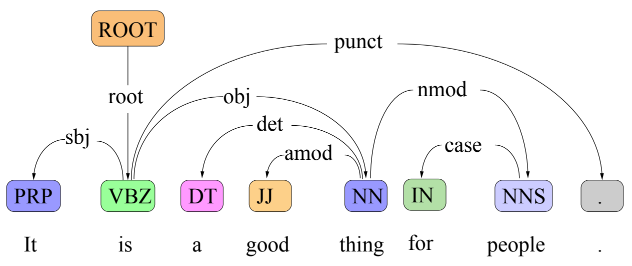

Figure 3. Dependency tree. One tree has only one root. One node except root in the dependency tree has only one parent and one dependency label.

Dependency parsing is to predict the existence and type of linguistic dependency relations between words in one sentences [39], which results in a dependency tree structure for every word in the sentence as shown in Fig. 3. Each word except for the root in one sentence has one and only one parent. The dependency label between the word and its parent can be viewed as a feature of this word which indicates the relationship type between it and its parent. For the root, it has no parent and thus no defined dependency label, we simply set its dependency label as **root**. POS tags can be processed in the same way as dependency label because each word in sentence is associated with one POS tag.

As a sentence is written in a linear way which only conveniently represents the relationship of neighboring elements, a syntactic parsing tree may disclose nonlinear relationship between words for the same sentence. The relationship between the word and its parent is represented as an edge. In the dependency tree shown in Fig. 3, word **It** is syntactically closer to **is** than **a** because there is a directed edge between **It** and **is**. As dependency relation represented as edges connects any word and its parent, and one word except root in sentence has one and only one parent in the dependency parse tree, we adopt position of parent node in the parse tree for each word as a syntactic feature. Besides, the depth in the parse tree for a node indicates how far it is from the root, which is the most salient word in a sentence, thus the tree depth indicates how important a word is in terms of syntactic impact, we then adopt tree depth of the node as another syntactic feature.

Note the parent position and tree depth of nodes are non-zero integer. The range of the two syntactic feature values are also limited to the length of sentence.

B. Adding Syntactic Features into Input Embedding

Linearization method has been adopted for integrating syntax into RNN-based NMT [12, 14]. In this work, following [14], we take three syntactic features, dependency label, POS tag and parent position respectively as three types of embeddings to feed the model. All these extra syntactic feature embeddings are concatenated to the original input embedding as the model input as shown in Fig. 2.

Viewing the dependency label and POS tag as two features, the model needs to individually put the features into a range of dimensions. To avoid the hidden vector divided by multi-head attention [7], the dimension of one head should be larger than the dimension of one feature.

Note that syntactic features introduced in this input embedding way can be updated during training, which is essentially important for effectively using parent position feature.

C. Syntactic Positional Encodings

The Transformer contains no recurrence and no convolution [7], in order to make use of word order information in sentence, it uses positional encoding (PE) to encode the position of words. The PE has the same dimension as the input embedding so that they can be summed. PE used in the Transformer is computed by

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{\text{model}}}), \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{\text{model}}}), \end{aligned} \tag{4}$$

where pos is word position in sentence, i is the dimension.

For each word, its parent position and tree depth in dependency parse tree are similar to the word position. Thus we design syntactic positional encoding (SPE) to accommodate two types of syntactic clues, the parent position and tree depth of the node. The functions to compute different dimensions of positional encoding are similar to the original Transformer

$$\begin{aligned} SPE_{(pos,2ni+b,c_f)} &= sin(pos/c_f^{2ni/d_{\text{model}}}), \\ SPE_{(pos,2ni+b+1,c_f)} &= cos(pos/c_f^{2ni/d_{\text{model}}}), \end{aligned} \tag{5}$$

where pos is the value of parent position or tree depth, n is the number of features, i is the dimension, b is the offset of dimensions and c_f is a constant for feature f. Equation (5) sets different features the same dimensions. Dimensions of different features are staggered by b. For different features, c_f will be set to different value. In detail, we add two SPE features, parent position and tree depth of node. For these two features, we set c_f to 2000 for the parent position and 400 for the tree depth. As the original PE is independent from SPE, it will be retained in our model.

D. Dependency Tree on Subwords

As we introduce a predicted syntactic parse tree which is built over a sequence of words while the current open-vocabulary NMT relies on a subword segmentation over source or target words which usually adopts BPE (byte pair encoding) algorithm [40], there is a word/subword mismatch when we try to integrate syntax into the source side of NMT models. For words segmented into multiple subwords, we set the first subword part of a word to inherit its dependency label, parent position and POS tag. For the rest subwords of the word, we build new edges between them and the first one, respectively. Note that our processing here is different from [14] that uses a feature to mark the boundary of subwords.

IV. Experiments

A. Datasets

We evaluate our model on two translation tasks, IWSLT14 German-English (De-En) and WMT14 English-German (En-De) .

IWSLT14 German-English IWSLT14 De-En dataset contains 153K training sentence pairs. We use 7K data from the training set as validation set and use the combination of dev2010, dev2012, tst2010, tst2011 and tst2012 as test set with 7K sentences which are preprocessed by script[2]. BPE algorithm is used to process words into subwords, and number of subword tokens in the shared vocabulary is 31K.

WMT14 English-German We use the WMT14 En-De dataset from Stanford[3] with 4.5M sentence pairs for training. We use the combination of newstest2012 and newstest2013 as validation set and newstest2014 as test set. The sentences longer than 80 are removed from the training dataset. Dataset is segmented by BPE so that number of subwords in the shared vocabulary is 32K.

Generally, we use Stanford Parser[4] to process German corpus to get dependency label and POS tags. For English corpus, we use Stanford Dependency Parser[5] to get dependency label and Stanford POS Tagger [6] to get predicted POS tags. The parent position of one word will be set to 0 if the parent is root.

B. Hyperparameters

The hyperparameters for our experiments are shown in Table IV. For De-En, we follow the setting of Transformer-small. For En-De, we follow the setting of Transformer-base. For both tasks, the dimension for one feature is 32. The input embedding size of our model is from summing up the dimensions of word embeddings and syntactic features.

C. Training

All our models are trained on one CPU (Intel i7-5960X) and one nVidia 1080Ti GPU. The implementation of model is based on OpenNMT-py[7]. We choose Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$ and the learning rate setting strategy, which are all the same as [7]. We use beam search decoder for De-En task with beam width 6. For En-De, following [7], the width for beam search is 5 and the length penalty α is 0.6. The batch size is 1024 for De-En and 2048 for En-De. We evaluate the translation results by using case-insensitive BLEU[8].

V. Results

Our baselines for De-En and En-De tasks are the Transformer-small and the Transformer-base. Table II is the results of our main experiments. For De-En and En-De tasks, our model which incorporates POS tags, parent position and dependency labels through embeddings respectively outperforms the baselines by 0.4 BLEU and 0.50 BLEU. Our model which incorporates POS tags and dependency labels through embeddings and

[2]https://github.com/eske/seq2seq/blob/master/config/IWSLT14/prepare-mixer.sh

[3]https://nlp.stanford.edu/projects/nmt/

[4]https://nlp.stanford.edu/software/lex-parser.html

[5]https://nlp.stanford.edu/software/nndep.html

[6]https://nlp.stanford.edu/software/tagger.html

[7]https://github.com/OpenNMT/OpenNMT-py/archive/0.7.0.zip

[8]https://github.com/OpenNMT/OpenNMT-py/blob/master/tools/multi-bleu.perl

Table II
BLEU SCORES ON IWSLT 2014 AND WMT14. THE BASELINES FOR DE-EN TASK AND EN-DE TASK ARE THE TRANSFORMER-SMALL AND THE TRANSFORMER-BASE, RESPECTIVELY.

Model	BLEU	
	DE-EN	EN-DE
Transformer (small)	31.80	-
Transformer (base)	-	26.80
Our method (POS Tag & Parent & Dependency label)	32.20	27.30
Our method (POS Tag & Dependency label & SPE)	31.90	27.15

Table III
BLEU SCORES ON WMT14 EN-DE EXPERIMENT WITH EMBEDDING TO ADD FEATURES. WE REMOVE DIFFERENT FEATURE, SET THE DIMENSIONS OF INPUT EMBEDDINGS TO 576 AND THE HEADS TO 9. IN THESE EXPERIMENTS, WE USE ONLY EMBEDDINGS TO INCORPORATE FEATURES.

Model	EN-DE
Transformer (base)	26.80
Our Model	27.30
- Parent Position	27.00
- POS Tags	27.01
- Dependency Label	26.70

Table IV
HYPERPARAMETERS FOR OUR EXPERIMENTS. FF IS SHORT FOR FEED-FORWARD LAYER. THE NUMBER OF HEADS IS BASED ON THE DIMENSION FOR WORD AND FEATURE.

Parameter	DE-EN		EN-DE	
	Without SPE	With SPE	Without SPE	With SPE
Layers	6	6	6	6
Dimension for Word	256	256	512	512
Dimension for Feature	32	32	32	32
Head	11	5	19	9
FF	1408	1280	2432	2304
Dropout	0.3	0.3	0.1	0.1

incorporates parent position and tree depth of nodes through SPE respectively outperforms the baselines by 0.10 BLEU and 0.35 BLEU. The results show that both of our model settings with syntax-informed features can enhance the respective NMT. The model using only embeddings to introduce syntax performs better than the model with SPE. We attribute this difference to the trainable ways between embedding and SPE. The input embedding can be updated during training but SPE cannot. Such representation updating difference causes the observable performance change of the models.

Table V
BLEU SCORES ON WMT14 EN-DE. WE RESPECTIVELY REMOVE THE FEATURE OF PARENT POSITION AND THE FEATURE OF TREE DEPTH FROM OUR MODEL WITH FULL SPE FEATURES.

Model	EN-DE
Transformer (base)	26.80
Our Model with SPE	27.15
- Parent Position	27.00
- Tree Depth	26.68

To compare importance of different features, we test models by removing one feature one by one from the three syntactic features. Table III shows that the model without the dependency labels obtains 26.70 BLEU score, while the BLEU scores without POS tags and parent position are respectively 27.01 and 27.00, which indicates that dependency label feature contributes most to the performance. Overall, the model with full three syntactic features outperforms the baseline model, which verifies the effectiveness of syntactic clues.

To evaluate the effect of SPE features, we perform experiments on WMT En-De 14 task by removing either of SPE features one by one. Table V shows the results. Comparing these results shows that the tree depth is more informative than the parent position for performance improvement. In the meantime, either of the two SPE features cannot individually outperform full SPE features which indicates either SPE feature is essential to the performance contribution.

VI. CONCLUSION

In this paper we propose to exploit syntactic information to enhance Transformer through input word embeddings and PE. The experimental results verify that the proposed methods using syntax can improve the performance of Transformer without changing the architecture of model. We also compare the proposed two ways of introducing syntax into the Transformer, which shows that embedding updating is a factor making the translation performance difference and thus the direct input embedding integration gives better translation. In addition, we empirically study the contribution of detailed syntactic features including POS tags, dependency labels, parent position and tree depth. We conclude that all the proposed syntactic features are indeed helpful for enhancing the Transformer in NMT.

REFERENCES

[1] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *EMNLP*, 2013, pp. 1700–1709.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014, pp. 3104–3112.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[4] Y. Wu and H. Zhao, "Finding better subword segmentation for neural machine translation," in *CCL*, 2018, pp. 53–64.

[5] H. Zhang and H. Zhao, "Minimum divergence vs. maximum margin: an empirical comparison on seq2seq models," in *ICLR*, 2019.

[6] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *ICML*, 2017, pp. 1243–1252.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 6000–6010.

[8] M. Galley, M. Hopkins, K. Knight, and D. Marcu, "What's in a translation rule?" in *HLT-NAACL*, 2004, pp. 273–280.

[9] M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer, "Scalable inference and training of context-rich syntactic translation models," in *ACL*, 2006, pp. 961–968.

[10] A. Eriguchi, K. Hashimoto, and Y. Tsuruoka, "Tree-to-sequence attentional neural machine translation," in *ACL*, 2016, pp. 823–833.

[11] R. Aharoni and Y. Goldberg, "Towards string-to-tree neural machine translation," in *ACL*, 2017, pp. 132–140.

[12] J. Li, D. Xiong, Z. Tu, M. Zhu, M. Zhang, and G. Zhou, "Modeling source syntax for neural machine translation," in *ACL*, 2017, pp. 688–697.

[13] X. Wang, H. Pham, P. Yin, and G. Neubig, "A tree-based decoder for neural machine translation," in *EMNLP*, 2018, pp. 4772–4777.

[14] R. Sennrich and B. Haddow, "Linguistic input features improve neural machine translation," in *ACL*, 2016, pp. 83–91.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[16] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[17] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *NAACL*, 2018, pp. 464–468.

[18] F. Xiao, J. Li, H. Zhao, R. Wang, and K. Chen, "Lattice-based transformer encoder for neural machine translation," in *ACL*, 2019, pp. 3090–3097.

[19] X. Ma, X. Zhang, H. Zhao, and B. Lu, "Dependency parser for Chinese constituent parsing," in *CIPS-SIGHAN*, 2010.

[20] X. Ma and H. Zhao, "Fourth-order dependency parsing," in *COLING*, 2012, pp. 785–796.

[21] H. Zhao, X. Zhang, and C. Kit, "Integrative semantic dependency parsing via efficient large-scale feature selection," *J. Artif. Intell. Res.*, vol. 46, pp. 203–233, 2013.

[22] Z. Li, S. He, Z. Zhang, and H. Zhao, "Joint learning of POS and dependencies for multilingual universal dependency parsing," in *CoNLL*, 2018, pp. 65–73.

[23] Z. Li, S. He, J. Cai, Z. Zhang, H. Zhao, G. Liu, L. Li, and L. Si, "A unified syntax-aware framework for semantic role labeling," in *EMNLP*, 2018, pp. 2401–2411.

[24] S. He, Z. Li, H. Zhao, and H. Bai, "Syntax for semantic role labeling, to be, or not to be," in *ACL*, 2018, pp. 2061–2071.

[25] J. Zhou and H. Zhao, "Head-driven phrase structure grammar parsing on penn treebank," in *ACL*, 2019, pp. 2396–2408.

[26] J. Cai, S. He, Z. Li, and H. Zhao, "A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware?" in *COLING*, 2018, pp. 2753–2765.

[27] Z. Li, S. He, H. Zhao, Y. Zhang, Z. Zhang, X. Zhou, and X. Zhou, "Dependency or span, end-to-end uniform semantic role labeling," in *AAAI*, 2019, pp. 6730–6737.

[28] C. Guan, Y. Cheng, and H. Zhao, "Semantic role labeling with associated memory network," in *NAACL-HLT*, 2019, pp. 3361–3371.

[29] X. Wang, D. Cai, L. Li, G. Xu, H. Zhao, and L. Si, "Unsupervised learning helps supervised neural word segmentation," in *AAAI*, 2019, pp. 7200–7207.

[30] D. Cai, H. Zhao, Z. Zhang, Y. Xin, Y. Wu, and F. Huang, "Fast and accurate neural word segmentation for Chinese," in *ACL*, 2017, pp. 608–615.

[31] D. Cai and H. Zhao, "Neural word segmentation learning for Chinese," in *ACL*, 2016, pp. 409–420.

[32] H. Zhao and C. Kit, "Integrating unsupervised and supervised word segmentation: The role of goodness measures," *Inf. Sci.*, vol. 181, no. 1, pp. 163–183, 2011.

[33] Z. Li, J. Cai, S. He, and H. Zhao, "Seq2seq dependency parsing," in *COLING*, 2018, pp. 3203–3214.

[34] H. Wang, H. Zhao, and Z. Zhang, "A transition-based system for universal dependency parsing," in *CoNLL*, 2017, pp. 191–197.

[35] Z. Zhang, H. Zhao, and L. Qin, "Probabilistic graph-based dependency parsing with convolutional neural network," in *ACL*, 2016, pp. 1382–1392.

[36] H. Li, Z. Zhang, Y. Ju, and H. Zhao, "Neural character-level dependency parsing for Chinese," in *AAAI*, 2018, pp. 5205–5212.

[37] Y. Wu, H. Zhao, and J. Tong, "Multilingual universal dependency parsing from raw text with low-resource language enhancement," in *CoNLL*, 2018, pp. 74–80.

[38] Z. Li, J. Cai, and H. Zhao, "Effective representation for easy-first dependency parsing," in *PRICAI*, 2019, pp. 351–363.

[39] X. Ma, Z. Hu, J. Liu, N. Peng, G. Neubig, and E. Hovy, "Stack-pointer networks for dependency parsing," in *ACL*, 2018, pp. 1403–1414.

[40] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *ACL*, 2016, pp. 1715–1725.