Syntax-based Transformer for Neural Machine Translation

Chunpeng Ma†, Akihiro Tamura††, Masao Utiyama†††, Eiichiro Sumita††† and Tiejun Zhao†

The Transformer (Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin 2017), which purely depends on attention mechanism, has achieved state-of-the-art performance on machine translation (MT). However, syntactic information, which has improved many previous MT models, has not been utilized explicitly by Transformer. We propose a syntax-based Transformer for MT, which incorporates source-side syntax structures generated by the parser into the self-attention and positional encoding of the encoder. Our method is general in that it is applicable to both constituent trees and packed forests. Evaluations on two language pairs show that our syntax-based Transformer outperforms the conventional (non-syntactic) Transformer. The improvements of BLEUs on English-Japanese, English-Chinese and English-German translation tasks are up to 2.32, 2.91 and 1.03, respectively. Furthermore, our ablation study and qualitative analysis demonstrate that the syntax-based self-attention does well in learning local structural information, while the syntax-based positional encoding does well in learning global structural information.

Key Words: Machine Translation, Deep Learning, Syntactic Information

1 Introduction

In recent years, neural machine translation (NMT) has been developing rapidly and has became the de facto standard approach for MT. Dominant NMT models follow the encoder-decoder paradigm. Based on networks used by the encoder/decoder, these models can be divided into several categories: convolutional neural network (CNN) based models (Gehring, Auli, Grangier, and Dauphin 2017), recurrent neural network (RNN) based models (Sutskever, Vinyals, and Le 2014), etc. Attention mechanism (Bahdanau, Cho, and Bengio 2014; Luong, Pham, and Manning 2015) plays an important role in these models. Vaswani et al. (2017) proposed the Transformer, which is based solely on attention mechanisms. Compared with CNN/RNN-based NMT models, Transformer has two major advantages: (1) it avoids the sequential dependencies in RNN-based NMT models, so that training can be parallelized and can be done efficiently, and (2) it uses self-attention mechanism and positional encoding, so that a wide range of context can be used.
Therefore, we focus on Transformer here.

Effectiveness of explicit usage of syntactic information has been proved by plenty of previous research. Syntactic information can be used in various models, including models in statistical machine translation (Mi and Huang 2008), RNN-based NMT (Kuncoro, Dyer, Hale, Yogatama, Clark, and Blunsom 2018) and CNN-based NMT (Watanabe, Tamura, and Ninomiya 2017). Syntactic information can be used on either the source-side (Eriguchi, Tsuruoka, and Cho 2017), or the target-side (Aharoni and Goldberg 2017), or both (Wu, Zhang, Zhang, Yang, Li, and Zhou 2018). Syntactic information can be represented as constituent trees (Eriguchi, Hashimoto, and Tsuruoka 2016), packed forests (Ma, Tamura, Utiyama, Zhao, and Sumita 2018), or graphs (Hashimoto and Tsuruoka 2017). There are also some attempts at using syntactic information explicitly in Transformer (Strubell, Verga, Andor, Weiss, and McCallum 2018). Their method utilized syntactic information from one single tree, while our proposed method is applicable to forests as well.

It is not straightforward to adapt RNN-based syntactic NMT methods for the Transformer, especially forest/graph-based methods, because of the inherent difference on network structures. Here, we propose a syntax-based Transformer for MT, which encodes syntactic structures (constituent trees or packed forests (Huang 2008)) on the source-side. In particular, we propose a syntax-based self-attention mechanism (Section 3.1) and syntax-based positional encoding (Section 3.2) for the encoder of the Transformer. The proposed syntax-based self-attention mechanism determines which words and/or constituent labels to pay attention to, on the basis of syntactic structures, rather than paying attention to all the words and constituent labels in the syntactic structure. The proposed syntax-based positional encoding encodes the syntactic distance between nodes in the syntactic structures, rather than encoding the absolute position (Vaswani et al. 2017) or relative word distance (Shaw, Uszkoreit, and Vaswani 2018).

This paper makes the following contributions.

- We propose a general syntax-based Transformer, which is applicable to both constituent trees and packed forests. It can encode large amount of syntactic information efficiently in parallel.
- Experiments on two datasets demonstrate the good performance of our syntax-based Transformer compared with the non-syntactic counterpart. This indicates the usefulness of using syntactic information explicitly.
- Qualitative analysis shows that our syntax-based Transformer learns syntactic characteristics successfully, and the proposed two modules learn syntactic information from different aspects.
2 Background

Different architectures in NMT, especially the Transformer, are reviewed in this section.

2.1 Architectures in NMT

Generally, models in NMT follow the encoder-decoder paradigm, which first encodes the input (plain sentences, syntactic structures, etc.) of the source language with a neural network (RNN or CNN, etc.) and generates the representation (a fixed-length vector or a sequence of vectors) of the input, and then generates the target sentence by decoding the representation.

The RNN-based NMT models are composed of two RNNs for the encoder and decoder, respectively. This kind of models has been adopted into industrial systems (Wu, Schuster, Chen, Le, Norouzi, Macherey, Krikun, Cao, Gao, Macherey, et al. 2016). Despite these success, the sequential nature of RNNs makes the models difficult to be trained in parallel. Therefore, the training process is very time-consuming.

The CNN-based NMT models stack multiple convolutional layers on both encoder and decoder (Gehring et al. 2017). Training can be fully parallelized and can be greatly accelerated by using GPUs. However, CNNs capture only limited context and thus long-term dependencies are difficult to learn, which may deteriorate the translation of long sentences.

The Transformer (Vaswani et al. 2017) is designed to address the disadvantages of the above two models. The encoder is a stack of layers. Each layer consists of two submodules: a self-attention network and a feed-forward network. The decoder is also a stack of layers. Compared with layers in the encoder, the decoder counterpart contains one more submodule: the encoder-decoder attention. At the bottom of the encoder and the decoder, positional encodings are added to the input embeddings. With the help of layer normalization (Ba, Kiros, and Hinton 2016) and residual connection (He, Zhang, Ren, and Sun 2016), the Transformer achieves state-of-the-art translation performance.

It should be noted that: (1) the Transformer uses self-attention mechanism in both encoder and decoder, and pays attention to all the inputs, so that long-term dependencies can be learned easily as RNN-based models, and (2) the Transformer encodes the positional information by using the positional encoding, so that training can be parallelized as CNN-based models. We incorporate syntactic information in both of them.
2.2 Self-attention Mechanism

Assume that there are $N$ layers in the encoder. For the $l$-th layer, denote the input and output sequence of the self-attention submodule as $x^{(l)} = (x_1^{(l)}, \ldots, x_n^{(l)})$ and $z^{(l)} = (z_1^{(l)}, \ldots, z_n^{(l)})$, respectively, where $n$ is the length of the input sequence and $l \in [1, N]$. For $i \in [1, n]$, we calculate $z_i^{(l)} \in \mathbb{R}^d$ as follows:

$$z_i^{(l)} = \sum_{j=1}^{n} \alpha_{ij}(x_j^{(l)}W_V), \quad (1)$$

$$\alpha_{ij}^{(l)} = \text{softmax}(\beta_{ij}^{(l)}), \quad (2)$$

$$\beta_{ij}^{(l)} = \frac{(x_i^{(l)}W_Q)(x_j^{(l)}W_K)^\top}{\sqrt{d}}, \quad (3)$$

where $W_V, W_Q, W_K \in \mathbb{R}^{d \times d}$ are parameter matrices. Note that each $z_i^{(l)}, i \in [1, n]$ is calculated from all the tokens in $(x_1^{(l)}, \ldots, x_n^{(l)})$.

Denote the source-side word sequence as $(w_1, \ldots, w_n)$, then $x_i^{(1)}$ is calculated as follows:

$$x_i^{(1)} = W_E[w_i], \quad (4)$$

where $W_E \in \mathbb{R}^{d \times |V|}$ is the embedding matrix and $|V|$ is the vocabulary size. $[\cdot]$ is an operator to get the specific column of a matrix, therefore $x_i^{(1)} \in \mathbb{R}^d$ is the $w_i$-th column of $W_E$.

The output sequence $(z_1^{(l)}, \ldots, z_n^{(l)})$ of the self-attention submodule is fed to the feed-forward submodule, and the output of the feed-forward submodule is fed to the next layer as $(x_1^{(l+1)}, \ldots, x_n^{(l+1)})$.

2.3 Positional Encoding

Different from RNN-based and CNN-based models, the self-attention mechanism itself is unable to encode the order of words in the input sequence. To make use of word order information, positional information (either absolute position or relative position) should be inserted into the model.

Vaswani et al. (2017) encodes absolute positional information by modifying Equation (4):

$$x_i^{(1)} = W_E[w_i] + PE(i),$$

$$PE(i)_{2k} = \sin(i/10000^{2k/d}), \quad (5)$$

$$PE(i)_{2k+1} = \cos(i/10000^{2k/d}),$$

where $2k$ and $2k + 1$ are even and odd dimension indices, respectively. They hypothesize the choice of sinusoidal function makes the model be able to learn relative positions, because of the periodicity of the function. This property is also shared by the relative positional encoding.
Ma et al. Syntax-based Transformer for Neural Machine Translation

introduced below.

On the other hand, Shaw et al. (2018) encodes relative positional information by modifying Equations (1) and (3) as follows, respectively:

\[
\begin{align*}
    z_i^{(l)} &= \sum_{j=1}^{n} \alpha_{ij} (x_j^{(l)}W^V + a_{ij}^V), \\
    \beta_{ij} &= \frac{(x_i^{(l)}W^Q)(x_j^{(l)}W^K + a_{ij}^K)^\top}{\sqrt{d}},
\end{align*}
\]

where \( a^K_{ij}, a^V_{ij} \in \mathbb{R}^d \) are vectors representing the relative positions between words. They are calculated as follows:

\[
\begin{align*}
    a^K_{ij} &= p^K_{\text{clip}(j-i,k)}, \\
    a^V_{ij} &= p^V_{\text{clip}(j-i,k)}, \\
    \text{clip}(m, k) &= \max(-k, \min(k, m)),
\end{align*}
\]

where \( k \) is the window size, \((p^K_{-k}, \ldots, p^K_k)\) and \((p^V_{-k}, \ldots, p^V_k)\) are relative position representations to be learned.

2.4 Syntax-based NMT

As for tree-based NMT models, a lot of different methods have been proposed. Trees can be used in either the source side (Li, Xiong, Tu, Zhu, Zhang, and Zhou 2017) or the target side (Aharoni and Goldberg 2017), or both (Wu et al. 2018), can be encoded either using tree-structured neural networks (Eriguchi et al. 2016) or with the help of linearization (Sennrich and Haddow 2016), and can be either constituent trees (Chen, Huang, Chiang, and Chen 2017) or dependency trees (Wu, Zhang, Yang, Li, and Zhou 2017). As for forest-based NMT models, Ma et al. (2018) is the first attempt, where linearized packed forests are encoded using RNNs in order to make the model robust to parsing errors. Zaremoodi and Haffari (2018) encode the packed forests using tree structured neural networks.

One important issue of syntax-based NMT models is the training efficiency. Since most of the models are based on RNNs, encoding input tokens sequentially, it is difficult to parallelize training. Furthermore, syntactic structures (especially packed forests) contain much more information than plain sentences. This aggravates the efficiency problem. On the contrary, the Transformer can be trained efficiently. However, as far as we know, syntactic information has never been used explicitly in the Transformer for MT. Therefore it is quite appealing to combine the advantages of both, which is the focus of this paper.

449
3 Methodology

We modify two modules of the encoder for the syntax-based Transformer: syntax-based self-attention mechanism (Section 3.1) and syntax-based positional encoding (Section 3.2). Syntactic information is incorporated in both of them.

Note that although our syntax-based Transformer receives a sequence (i.e., a linearized sequence of a syntax structure) as a conventional Transformer does, our syntax-based Transformer is independent of the linearization algorithm used for generating the input sequence. This is a remarkable difference from previous syntax-based NMT models based on linearization (Li et al. 2017; Chen et al. 2017). Since there is no evidence for the preference of any linearization methods, it is more reasonable to propose an encoding method which is equivalent to them all.

Only the encoder is involved in the proposed method, while the decoder is identical to the baseline Transformer.

3.1 Syntax-based Self-attention

Since a constituent tree can be regarded as a specific instance of a packed forest, we first give the description of our method based on packed forests.

Given the source-side word sequence \( (w_1, \ldots, w_n) \), the parser generates the packed forest \( \mathcal{F} = (V, E) \). We assume that the nodes in \( V \) are sorted such that all words are located in front of constituent labels, i.e., \( V = (w_1, \ldots, w_n, v_1, \ldots, v_m) \). Here, \( w_i (i \in [1, n]) \) is a word, and \( v_i (i \in [1, m]) \) is a constituent label accompanied with its span, which means that forest nodes like “NP[2,3]” and “NP[2,4]” are different. \( (v_1, \ldots, v_m) \) is sorted by some kind of sorting algorithm (e.g., top-down style linearization (Vinyals, Kaiser, Koo, Petrov, Sutskever, and Hinton 2015) or bottom-up style linearization (Ma, Liu, Tamura, Zhao, and Sumita 2017), etc.). \(^1\) \( E = \{e_1, \ldots, e_l\} \) is the set of hyperedges, and each \( e_i (i \in [1, l]) \) is a three-tuple \( e_i = (H_i, T_i, S_i) \), where \( H_i, T_i \in V^* \) are heads and tails of \( e_i \), and \( S_i \in \mathbb{R}_{\leq 0} \) is the score given by the parser, reflecting the confidence of the parser to the hyperedge.\(^2\) The score is the logarithm of a probability, therefore it is non-positive.

---

\(^1\) Different ordering of words and constituent labels correspond to different input sequences, which further influence the final output.

\(^2\) Although the original definition of a head in the packed forest is \( H_i \in V \), the parser released by Huang (2008), who defines the original packed forests, permits multiple parents for each node. Therefore, we follow the extended definition (i.e., \( H_i \in V^* \)).
Then, $(x^w_1, \ldots, x^w_n, x^v_1, \ldots, x^v_m)$ is calculated:

$$x^w_i = W^{Ew}[w_i], \quad x^v_i = W^{Ev}[v_i],$$  \hspace{1cm} (11)

where $W^{Ew} \in \mathbb{R}^{d \times |V^w|}, W^{Ev} \in \mathbb{R}^{d \times |V^v|}$ are embedding matrices. $V^w$ and $V^v$ are source language dictionary and constituent label dictionary, respectively. $|V^w|$ is the vocabulary size of the source language, and $|V^v|$ is the total number of constituent labels. For convenience, we denote $X^w = (x^w_1, \ldots, x^w_n)$ and $X^v = (x^v_1, \ldots, x^v_m)$.

Then, we calculate the sequence $(z^w_1, \ldots, z^w_n, z^v_1, \ldots, z^v_m)$ on the basis of the source-side syntactic information given by the constituent tree or packed forest. In particular, rather than using Equation (3), we compute $\beta_{ij}$ as follows:

$$\beta_{ij} = \begin{cases} 
\frac{1}{\sqrt{d}} (x^Q_iW^Q)(x^K_jW^K)^\top, & x_j \in \mathcal{N}(x_i), \\
-\infty, & x_j \notin \mathcal{N}(x_i), 
\end{cases}$$  \hspace{1cm} (12)

where $x_i, x_j \in X^w \cup X^v$. The function $\mathcal{N}(x)$ returns the syntax-based neighborhood of $x$. Intuitively, only the words (constituent labels) which are near in the tree or packed forest should be paid attention to, and the other words (constituent labels) are safe to be ignored. The syntax-based neighborhood function is as follows:

$$\mathcal{N}(x) = \begin{cases} 
X^w \cup \mathcal{N}^\downarrow(x, d^\downarrow), & x \in X^w, \\
\mathcal{N}^\uparrow(x, d^\uparrow) \cup \mathcal{N}^\downarrow(x, d^\downarrow), & x \in X^v, 
\end{cases}$$  \hspace{1cm} (13)

where $d^\downarrow$ and $d^\uparrow$ are the hyperparameters reflecting the syntactic depths in top-down and bottom-up directions, respectively. For words, we assume that all the words are useful, and syntactic neighbors are also useful. For constituent labels, we assume that only syntactic neighbors are useful. The functions $\mathcal{N}^\uparrow(x, d)$ and $\mathcal{N}^\downarrow(x, d)$ are syntactic neighbors in the bottom-up and top-down directions, respectively, and are defined recursively as follows:

$$\mathcal{N}^\uparrow(x, d) = x \cup \left( \bigcup_{y \in \mathcal{T}(x)} \mathcal{N}^\uparrow(y, d - 1) \right),$$  \hspace{1cm} (14)

$$\mathcal{N}^\downarrow(x, d) = x \cup \left( \bigcup_{y \in \mathcal{H}(x)} \mathcal{N}^\downarrow(y, d - 1) \right),$$  \hspace{1cm} (15)

---

3 For simplicity of notations, henceforth, we omit the superscript $(l)$, the index of layers in the encoder.

4 For simplicity of notations, we discriminate neither $x^w_i$ and $w_i$, nor $x^v_i$ and $v_i$, for the case with no ambiguity.
and the terminal conditions are:
\[
\mathcal{N}^\uparrow(x, 0) = \mathcal{N}^\downarrow(x, 0) = \{x\},
\]
\[
\mathcal{N}^\uparrow(\emptyset, d) = \mathcal{N}^\downarrow(\emptyset, d) = \emptyset, \quad \forall d \in \mathbb{Z}_+.
\]

\(\mathcal{H}(x)\) and \(\mathcal{T}(x)\) are the parents set and children set of \(x\), respectively. They are defined as follows:
\[
\mathcal{H}(x) = \{v \in V \mid \exists (\mathcal{H}, \mathcal{T}, \mathcal{S}) \in \mathcal{E}, x \in \mathcal{T} \land v \in \mathcal{H}\},
\]
\[
\mathcal{T}(x) = \{v \in V \mid \exists (\mathcal{H}, \mathcal{T}, \mathcal{S}) \in \mathcal{E}, x \in \mathcal{H} \land v \in \mathcal{T}\}.
\]

Therefore, the top-down syntactic neighborhood of a node \(x\) should consist of \(x\) itself and all the ancestors within depth \(d^\downarrow\), and similarly for the bottom-up syntactic neighborhood. Note that for constituent trees, there is only one hyperedge connected to one node. In this way, syntactic information is incorporated into the calculation of the self-attention.

Furthermore, as demonstrated by Ma et al. (2018), scores are important for the performance of syntax-based NMT systems. To utilize the scores, we calculate the attention by Equation (20) rather than Equation (2), which is similar to the Score-on-Attention model of Ma et al. (2018).
\[
\alpha_{ij} = \begin{cases} 
  s(x_i, x_j) \cdot \text{softmax}(\beta_{ij}), & x_j \in \mathcal{N}(x_i), \\
  0, & x_j \notin \mathcal{N}(x_i), 
\end{cases}
\]
where \(s(x_i, x_j)\) is calculated as follows:
\[
s(x_i, x_j) = \sum_{E_k \in \mathcal{H}(x_i, x_j) \cup \mathcal{T}(x_i, x_j)} \prod_{(\cdots, \mathcal{S}_l \rangle \in E_k} \exp(\mathcal{S}_l),
\]
where the functions \(\mathcal{H}_c\) and \(\mathcal{T}_c\) return the sets of hyperedge sequences, and are defined as follows:
\[
\mathcal{H}_c(x, y) = \{(\mathcal{H}_1, \mathcal{T}_1, \mathcal{S}_1), \ldots, (\mathcal{H}_t, \mathcal{T}_t, \mathcal{S}_t) \mid x \in \mathcal{T}_1 \land y \in \mathcal{H}_t \land (\mathcal{H}_1, \mathcal{T}_1, \mathcal{S}_1) \in \mathcal{E} \\
\land (\forall i \in \mathbb{Z} \cap [2, t], (\mathcal{H}_i, \mathcal{T}_i, \mathcal{S}_i) \in \mathcal{E} \land \mathcal{H}_{i-1} \cap \mathcal{T}_i \neq \emptyset)\},
\]
\[
\mathcal{T}_c(x, y) = \{(\mathcal{H}_1, \mathcal{T}_1, \mathcal{S}_1), \ldots, (\mathcal{H}_t, \mathcal{T}_t, \mathcal{S}_t) \mid x \in \mathcal{H}_1 \land y \in \mathcal{T}_t \land (\mathcal{H}_1, \mathcal{T}_1, \mathcal{S}_1) \in \mathcal{E} \\
\land (\forall i \in \mathbb{Z} \cap [2, t], (\mathcal{H}_i, \mathcal{T}_i, \mathcal{S}_i) \in \mathcal{E} \land \mathcal{T}_{i-1} \cap \mathcal{H}_i \neq \emptyset)\},
\]
where \(t\) is the length of the sequence. Intuitively, each element in \(\mathcal{H}_c\) (or \(\mathcal{T}_c\)) is a path between node \(x\) and \(y\) in the bottom-up (or top-down) direction. In this way, scores are utilized by the syntax-based self-attention mechanism.
3.2 Syntax-based Positional Encoding

The syntax-based self-attention mechanism does not capture the position of each node in the constituent tree/packed forest. Therefore, we propose a syntax-based positional encoding for directly encoding syntactic positional information. We can adapt the absolute/relative positional encoding to the syntax-based machine translation model straightforwardly. We first linearize the constituent tree/packed forest, and calculate the positional encoding of each word/label based on its position in the sequence. However, this naive adaptation of the absolute/relative positional encoding depends on the order of nodes in the sequence, while the optimal syntactic sequence for NMT is an open problem. Therefore, we propose a syntax-based positional encoding that relies only on pure syntactic structures (constituent trees/packed forests) and does not depend on the linearization method of the syntactic structures.

The proposed syntax-based positional encoding is defined by modifying Equation (1) and (3) as follows:

\[
\begin{align*}
    z_i &= \sum_{x_j \in N(x_i)} \alpha_{ij}(x_jW^V + b^V_{ij}), \\
    \beta_{ij} &= \begin{cases} 
        \frac{1}{\sqrt{d}}(x_iW^Q)(x_jW^K + b^K_{ij})^\top, & x_j \in N(x_i), \\
        -\infty, & x_j \notin N(x_i),
    \end{cases}
\end{align*}
\]

(24) (25)

where \(b^K_{ij}, b^V_{ij} \in \mathbb{R}^d\) are two kinds of distributed representation of syntactic distance between two nodes \(x_i\) and \(x_j\). They are calculated as follows:

\[
\begin{align*}
    b^K_{ij} &= \begin{cases} 
        q^w_{dist(i,j)}, & x_i \in X^w, \\
        q^K_{dist(i,j)}, & x_i \in X^v,
    \end{cases} \\
    b^V_{ij} &= \begin{cases} 
        q^w_{dist(i,j)}, & x_i \in X^w, \\
        q^V_{dist(i,j)}, & x_i \in X^v.
    \end{cases}
\end{align*}
\]

(26)

Here, every \(q\) is the distributed representation of syntactic distance to be learned. Similar to \(p\) in Equation (8) and (9), they are vectors in \(\mathbb{R}^d\). \(\text{dist}(i,j)\) is defined as follows:

\[
\text{dist}(i,j) = \text{mode}\left(\{|E| : E \in \mathcal{H}_c(x_i, x_j) \cup \mathcal{V}_c(x_i, x_j)\}\right),
\]

(27)

where \(\text{mode}(\cdot)\) returns the element that appears most often in the argument set.\(^5\) Intuitively, \(\text{dist}(i,j)\) can be regarded as the most frequent path length between \(x_i\) and \(x_j\) in two directions according to the syntactic structures. According to their definitions, \(d^\uparrow\) and \(d^\downarrow\) are the maximum

\(^5\) When several different elements meet the condition, the minimum one is returned.
syntactic distances in two directions, respectively, while dist\((i, j)\) is the syntactic distance of the most frequent path. Therefore, we can see that

\[
\max_{i,j} \text{dist}(i,j) \leq \max(d^\uparrow, d^\downarrow) \equiv d^\uparrow \forall i, j, \quad (28)
\]

therefore our syntax-based positional encodings to be learned is a \(4 \times (d^\uparrow + 1)\) weight matrix:

\[
Q \equiv \begin{pmatrix}
q^{wK}_0 & \ldots & q^{wK}_{d^\uparrow} \\
q^{wV}_0 & \ldots & q^{wV}_{d^\uparrow} \\
q^{vK}_0 & \ldots & q^{vK}_{d^\uparrow} \\
q^{vV}_0 & \ldots & q^{vV}_{d^\uparrow}
\end{pmatrix},
\quad (29)
\]

where empty elements may exist.

Note that the values of \(b^K_{ij}\) and \(b^V_{ij}\) are independent of the orders of elements in \(X^v\) (i.e., different linearization methods of trees/forests).

4 Experiments

4.1 Configuration

To evaluate the effectiveness of our syntax-based Transformer models, we did the experiments on three language pairs: English(En)-to-Japanese(Ja), English(En)-to-Chinese(Zh) and English(En)-to-German(De).\(^6\) The sizes of training corpora are roughly \(10^5, 10^6, 10^7\) sentence pairs, respectively, so that we can investigate the effects with respect to the scales of training corpora. ASPEC corpus is introduced in Nakazawa, Yaguchi, Uchimoto, Utiyama, Sumita, Kurohashi, and Isahara (2016). Only 100k high-quality sentence pairs were used in our experiments. LDC corpus is composed of the following parts: LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08, and LDC2005T06. WMT 2019 (En-De) is preprocessed following the steps in Ng, Yee, Baevski, Ott, Auli, and Edunov (2019) and 10M sentences are randomly chosen.

The constituent trees and packed forests of English sentences are obtained by the constituent parser proposed by Huang (2008).\(^7\),\(^8\) The parser converts a tree/packed forest to a sequence where

\(^6\) We choose English as the source language in all experiments, because high-quality packed forests can be obtained. Note that the proposed methods are applicable to any source language, as long as the packed forests are available.

\(^7\) http://web.engr.oregonstate.edu/~huanlian/software/forest-reranker/forest-charniak-v0.8.tar.bz2

\(^8\) We also confirmed the effectiveness of our models when using the Egret parser (https://github.com/neubig/egret), and found little difference in experimental results.
all words are located in front of constituent nodes and all constituent nodes are located in front of their parents, which is the input of our syntax-based Transformer. Forests whose total numbers of words and constituent nodes are larger than 150 are filtered out. To make the comparison fair, we also filtered out these long sentences for other state-of-the-art models. Chinese word segmentation is done by the Stanford segmentation tool.\footnote{https://nlp.stanford.edu/software/stanford-segmenter-2017-06-09.zip} For Japanese sentences, we followed the preprocessing steps recommended in WAT 2017.\footnote{http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2017/baseline/dataPreparationJE.html}

We implemented our framework based on \textit{OpenNMT}\footnote{http://opennmt.net/} (Klein, Kim, Deng, Senellart, and Rush 2017). $d^\uparrow$ and $d^\downarrow$ are tuned on the development data and the other hyperparameters are chosen as recommended by the Transformer model implemented by \textit{OpenNMT}. Both the encoder and the decoder have 6 layers. The dimensions of hidden vectors and word embeddings are 512. The multi-head attention has 8 heads, and the dropout probability is 0.1.

The vocabulary sizes for source language and target language are both 50,000. Because it is difficult to obtain subword-based packed forests, all the experiments are word-based. As for optimization, we used the Adam optimizer (Kingma and Ba 2014), with $\beta_1 = 0.9$, $\beta_2 = 0.998$, and $\epsilon = 10^{-9}$. Warmup and decay strategy for learning rate are also used, with 8,000 warmup steps. We also used the label smoothing strategy (Szegedy, Vanhoucke, Ioffe, Shlens, and Wojna 2016) with $\epsilon_{ls} = 0.1$. The number of training epochs is fixed to 50, and we selected the best model on the development set for testing.

\begin{table}[h]
\centering
\caption{Statistics of the corpora}
\begin{tabular}{llc}
\hline
Language & Corpus & Usage & \#Sent. \\
\hline
En-Ja & ASPEC & train & 100,000 \\
 & & dev. & 1,790 \\
 & & test & 1,812 \\
 & LDC & train & 1,203,345 \\
En-Zh & NIST MT 02 & dev. & 876 \\
 & NIST MT 03 & test & 919 \\
 & NIST MT 04 & & 1,788 \\
 & NIST MT 05 & & 1,082 \\
En-De & WMT 2019 & train & 10,000,000 \\
 & newestest 2017 & dev. & 3,004 \\
 & newestest 2018 & test & 2,998 \\
\hline
\end{tabular}
\end{table}
4.2 Experimental Results

Table 2 summarizes the experimental results of our models and compares our models with previous state-of-the-art models. As stated in Section 3.1, for all syntax-based configurations, constituent trees/packed forests are linearized as \( V = (w_1, \ldots, w_n, v_1, \ldots, v_m) \), i.e., all constituent labels are located after all words. See Figure 3 for an illustration. Furthermore, absolute positional encoding is adapted to each models straightforwardly (see Section 3.2 for details). “Line Input” means that the model structure is identical to the plain Transformer (Vaswani et al. 2017), while the source-side input is linearized trees/forests. “Syntax SA” is the proposed model that uses the syntax-based self-attention mechanism (Section 3.1), and “+ Syntax PE” is the proposed model that uses the syntax-based positional encodings (Section 3.2) in addition to the syntax-based self-attention mechanism. Results of Syntax-based RNN systems are extracted from Ma et al. (2018), while Plain Transformer is implemented on OpenNMT. The translation performance is evaluated by character-level BLEU score (Papineni, Roukos, Ward, and Zhu 2002) for En-Ja and En-Zh, and by SacreBLEU score (Post 2018) for En-De. The brevity penalties (BP) are identical for all language pairs.

We can see that the results of “Line Input” are all worse than plain Transformers, even though syntactic information is encoded straightforwardly in the input. This means that the Transformer failed to learn syntactic information automatically with the naive linearized input. This is because

| Type                  | Model    | En-Ja MT 03 | En-Ja MT 04 | En-Ja MT 05 | En-Ja MT 05 | En-Zh MT 03 | En-Zh MT 04 | En-Zh MT 05 | En-Zh MT 05 | En-De MT 03 | En-De MT 04 | En-De MT 05 | En-De MT 05 |
|-----------------------|----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Syntax-based RNN      | Eriguchi+ 16 | 37.52       | 29.71       | 31.56       | 30.33       | 40.31       | 31.35       | 33.14       | 31.23       | 43.19       | 44.28       | 45.35       | 46.43       |
|                       | Chen+ 17 | 36.94       | 29.64       | 31.25       | 29.59       | 39.12       | 31.35       | 33.14       | 31.23       | 43.19       | 44.28       | 45.35       | 46.43       |
|                       | Li+ 17 | 36.21       | 29.60       | 31.96       | 29.84       | 39.55       | 31.35       | 33.14       | 31.23       | 43.19       | 44.28       | 45.35       | 46.43       |
|                       | Ma+ 18 | **42.17**   | 31.35       | 33.14       | 31.23       | 41.39       | 31.35       | 33.14       | 31.23       | 41.39       | 43.19       | 45.35       | 46.43       |
| Plain Trans.          | Vaswani+ 17 | 34.09       | 33.85       | 34.84       | 32.28       | 44.32       | 33.85       | 34.84       | 32.28       | 44.32       | 45.35       | 46.43       | 47.45       |
|                       | Shaw+ 18 | 34.42       | 34.26       | 35.29       | 32.81       | 43.96       | 34.26       | 35.29       | 32.81       | 43.96       | 45.35       | 46.43       | 47.45       |
| Syntax Trans. (Tree)  | Line Input | 33.59       | 33.77       | 34.55       | 31.41       | 42.31       | 33.77       | 34.55       | 31.41       | 42.31       | 43.82       | 45.35       | 46.43       |
|                       | Syntax SA | 35.53\textsuperscript{+} | 35.40\textsuperscript{+} | 35.89\textsuperscript{+} | 33.00\textsuperscript{†} | 44.91\textsuperscript{†} | 35.53\textsuperscript{+} | 35.40\textsuperscript{+} | 35.89\textsuperscript{+} | 33.00\textsuperscript{†} | 44.91\textsuperscript{†} | 45.18\textsuperscript{†} | 46.35\textsuperscript{†} |
|                       | + Syntax PE | **36.76\textsuperscript{+}** | 36.40\textsuperscript{+} | 34.58\textsuperscript{+} | 34.58\textsuperscript{+} | 45.18\textsuperscript{†} | **36.76\textsuperscript{+}** | 36.40\textsuperscript{+} | 34.58\textsuperscript{+} | 34.58\textsuperscript{+} | 45.18\textsuperscript{†} | 46.35\textsuperscript{†} | 47.55\textsuperscript{†} |
| Syntax Trans. (Forest)| Line Input | 34.62       | 31.10       | 32.48       | 30.25       | 41.84       | 31.10       | 32.48       | 30.25       | 41.84       | 43.82       | 45.35       | 46.43       |
|                       | Syntax SA | 35.37\textsuperscript{†} | 34.85\textsuperscript{†} | 35.06       | 33.24\textsuperscript{†} | 45.23\textsuperscript{†} | 34.85\textsuperscript{†} | 35.06       | 33.24\textsuperscript{†} | 45.23\textsuperscript{†} | 45.35\textsuperscript{†} | 46.43       | 47.55       |
|                       | + Syntax PE | 36.41\textsuperscript{†} | 36.18\textsuperscript{†} | **36.93\textsuperscript{†}** | 34.31\textsuperscript{†} | 45.35\textsuperscript{†} | **36.93\textsuperscript{†}** | 34.31\textsuperscript{†} | 45.35\textsuperscript{†} | 45.35\textsuperscript{†} | 46.43       | 47.55       | 48.65       |

Paired bootstrap resampling significance test has been done. \(*: \text{significantly better w.r.t. Vaswani+ 17.}\) \(\dagger: \text{significantly better w.r.t. "Line Input".} \) \(p < 0.05. \) Values of \((d^2, d')\) are \((0, 1), (1, 1), (0, 1)\) for En-Ja, En-Zh, En-De, respectively, which are tuned on the development set.
the input sequence is quite long, which is problematic for the Transformer (Dai, Yang, Yang, Cohen, Carbonell, Le, and Salakhutdinov 2019). In contrast, “Syntax SA” outperforms “Line Input”, and moreover “+Syntax PE” is better than “Syntax SA”. This indicates that both the proposed syntax-based self-attention mechanism and the syntax-based positional encoding improve the translation performance significantly.

The improvement of performance on En-to-De is smaller than the improvements on En-to-Ja and En-to-Zh. This is because fairly enough syntactic information has already been captured by the large training corpus. However, the proposed methods still help improving the BLEU scores, and the improvement is statistical significant. “Line Input” is significantly worse than the baseline, while the difference between “Syntax SA” and “+Syntax PE” is not significant.

We find that there is a gap of performance between RNN-based models and Transformer models. Although the number of sentences and target languages are different among the corpora we used, we still found some interesting facts. Our syntax-based Transformer achieves the best performance on En-to-Zh and En-to-De tasks, although ours is worse than RNN-based forest NMT models on En-to-Ja task. This indicates that there are inherent differences between RNN-based models and Transformer models w.r.t. different sizes of datasets. RNN-based models are more robust to small-scale datasets, while Transformer performs well for large-scale datasets. Furthermore, difficulty in hyperparameter tuning is different, especially when syntactic information are utilized. Even if Ma et al. (2018) succeed to incorporate packed forests into an RNN-based NMT model, it is far from trivial to incorporate packed forests into the Transformer. Previous studies (Lakew, Cettolo, and Federico 2018) reported that the configuration of hyperparameters of the Transformer is much more tricky than that of the RNN-based NMT models. However, we simply use the same hyperparameters on all language pairs in our experiments.

Despite the relatively worse BLEU score on En-to-Ja task, our system is much faster. On En-to-Ja tasks, it takes about 22 hours with one GPU to train our model, while for the system of Ma et al. (2018), the training time is about 240 hours (11 times slower than ours).

5 Discussion

5.1 Hyperparameter Tuning and Ablation Studies

We use ASPEC corpus for hyperparameter tuning and ablation studies, because of its small size. The results are summarized in Table 3 and 4.
5.1.1 Influence of \(d^\uparrow\) and \(d^\downarrow\)

Table 3 shows the result of adjusting \(d^\uparrow\) and \(d^\downarrow\).\(^{12}\) We can see that the best \(d^\uparrow\) is 0, and the best \(d^\downarrow\) is 1 and 2 for the case of using constituent trees and packed forests, respectively. If \(d^\uparrow\) and \(d^\downarrow\) are too large, then too much noise is involved. The difference of optimal values of \(d^\uparrow\) and \(d^\downarrow\) indicates different effects of parents and children in a syntax tree (forest). Parents bring more noise and less information into the model, because the global coarse structure of different sentences are similar. For example, almost all English sentences consist of the subject, the verb, and the object.

5.1.2 Influence of scores

Comparing the first two rows in Table 4, we can see that for both constituent trees and packed

\(^{12}\)When \(d^\uparrow = d^\downarrow = 0\), all terminals are neighbors to each other, and non-terminals have no neighbors except themselves. In this case, syntactic attention scores are zeros, so that syntactic information are blocked, although syntactic labels indeed appeared in the input sentence.
Ma et al. Syntax-based Transformer for Neural Machine Translation

forests, using scores improves the translation performance. Note that we used absolute positional encoding instead of syntax-based positional encoding, so we can conclude that the improvement on performance was completely from syntax scores. This indicates that scores are important for the syntax-based Transformer, as was the case with the syntax-based NMT based on RNNs as demonstrated in Ma et al. (2018).

5.1.3 Effect of different positional encodings

Comparing the 2nd, 3rd, and 4th rows of Table 4, we can see that for the syntax-based Transformer, $b^K_{ij}$ is useful while $b^V_{ij}$ decreases translation performance significantly. This might follow the conclusion by Shaw et al. (2018) that positional encodings on keys (i.e., $a^K_{ij}$ in Equation (7)) are useful while positional encodings on values (i.e., $a^V_{ij}$ in Equation (6)) are useless.

By comparing the 2nd, 3rd, 5th, and 6th rows of Table 4, we can see that without any positional encodings, the performance is quite bad. Using either of positional encodings improves the performance significantly, and the combination of absolute positional encoding and $b^K_{ij}$ performs the best.

5.1.4 Tree or forest?

Although Ma et al. (2018) concludes that for RNN-based NMT systems, using packed forests is definitely better than using constituent trees, we found that it is not necessarily true for our syntax-based Transformer. For some configurations, using forests performs worse than using trees. For different models, noises in the forest affect the performance in different manners.

5.2 Qualitative Analysis

Figure 1 demonstrates the translation results of an English sentence, where only “+Syntax PE” correctly translates the source sentence. Figure 3 shows the self-attention learned by the encoders of different models, where attentions with small values are filtered out. For the cases of “Plain Transformer” and “Line Input”, only few lines are left. Even for these lines, the quality\(^\text{13}\) is still quite bad. For example, in Figure 3(a), “many” pays attention to “sprinkler”, and in Figure 3(b), “There” pays attention to “which”. These do not make any sense. Furthermore,

\(^{13}\text{One may wonder the meaning of “quality”. Here, when we say the quality of the attention mechanism is high, we mean that a high attention score is assigned to semantically/syntactically closely related words/labels, and vice versa. This argument is supported by many previous researches (Bahdanau et al. 2014; Luong et al. 2015; Vaswani et al. 2017), where they show the attention matrices and illustrate the similarity of attention matrices and word alignments. Therefore, it is reasonable to interpret the attention mechanism as the strength of connections between words/labels, and a high-quality attention mechanism should capture the connection to some extent.}\)
There are many semiconductor factories which have not installed the sprinkler, even if the fire alarm has been installed.

In contrast, attentions learned by the encoders of syntax-based Transformers are much better. Using syntax-based self-attention mechanism (Figure 3(c)), although we set $d^u = d^v = \infty$, the encoder learns to pay more attention to neighborhood nodes. For example, “[12] DT” in the key sequence are paid attention to by two nodes in the query sequence: “the” and “[25] NP”. These are the child and parent, respectively. On the other hand, the grandparent “[32] VP” is not paid attention to.

When syntax-based positional encoding is also used (Figure 3(d)), the connections become much stronger and much clearer. Unlike the many-to-many connections in Figure 3(c), most of the connections in Figure 3(d) are one-to-one. Part-of-speech tags pay attention to the corresponding words, and words in the query pay attention to the corresponding ancestor nodes near the root.

Comparing Figure 3(c) and Figure 3(d), we can conclude that syntax-based self-attention is good at learning local syntax structures, while syntax-based positional encoding is good at learning global syntax structures.
6 Conclusion and Future Works

We proposed a syntax-based Transformer. It combines the advantages of the Transformer and syntax-based NMT system, two state-of-the-art systems in machine translation. It can utilize rich syntactic information from constituent trees or packed forests, meanwhile can be trained efficiently in parallel. It consists of two important modules: syntax-based self-attention mechanism and syntax-based positional encodings. Both of them encode syntactic information, and improve the translation performance. The proposed syntax-based Transformer outperforms the RNN-based syntax NMT models on En-to-Zh tasks, and outperforms the non-syntactic Transformer baseline.
Fig. 3  Self-attention of the last encoder layer. Upper sequences are keys and lower sequences are queries. We set $d^k = d^v = \infty$, to let the models determine by themselves how much syntactic information should be used. Attentions with values less than 0.2 are filtered out. Darker line indicates larger attention. In (b), on the right side of "[3] VBP", there are no lines, so we omit this part for clarity. The corresponding constituent tree is shown in Figure 2. Nodes are indexed according to the linearization order.

on all language pairs. However, too much syntactic information may import noise, and forests are not necessarily better than trees. Even when the training corpus is quite large, the proposed method is effective although the improvement becomes small. Qualitative analysis indicates that syntax-based self-attention and syntax-based positional encodings learn different aspects of syntax structure. In future, we would like to incorporate target-side syntax structures into our model.

Reference

Aharoni, R. and Goldberg, Y. (2017). “Towards String-To-Tree Neural Machine Translation.” In Proceedings of ACL-2017, pp. 132–140.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). “Layer Normalization.” arXiv preprint arXiv:1607.06450.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). “Neural Machine Translation by Jointly Learning
Ma et al. Syntax-based Transformer for Neural Machine Translation

to Align and Translate. “arXiv preprint arXiv:1409.0473.
Chen, H., Huang, S., Chiang, D., and Chen, J. (2017). “Improved Neural Machine Trans-
lution with a Syntax-Aware Encoder and Decoder.” In Proceedings of ACL-2017 (1),
pp. 1936–1945.
Dai, Z., Yang, Z., Yang, Y., Cohen, W. W., Carbonell, J., Le, Q. V., and Salakhutdinov, R.
(2019). “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context.”
arXiv preprint arXiv:1901.02860.
Eriguchi, A., Hashimoto, K., and Tsuruoka, Y. (2016). “Tree-to-Sequence Attentional Neural
Machine Translation.” In Proceedings of ACL-2016 (1), pp. 823–833.
Eriguchi, A., Tsuruoka, Y., and Cho, K. (2017). “Learning to Parse and Translate Improves
Neural Machine Translation.” In Proceedings of ACL-2017 (2), pp. 72–78.
Gehring, J., Auli, M., Grangier, D., and Dauphin, Y. (2017). “A Convolutional Encoder Model
for Neural Machine Translation.” In Proceedings of ACL-2017 (1), pp. 123–135.
Hashimoto, K. and Tsuruoka, Y. (2017). “Neural Machine Translation with Source-Side Latent
Graph Parsing.” In Proceedings of EMNLP-2017, pp. 125–135.
He, K., Zhang, X., Ren, S., and Sun, J. (2016). “Deep Residual Learning for Image Recognition.”
In Proceedings of CVPR-2016, pp. 770–778.
Huang, L. (2008). “Forest Reranking: Discriminative Parsing with Non-Local Features.” In
Proceedings of ACL-2008 (1), pp. 586–594.
Kingma, D. P. and Ba, J. (2014). “Adam: A Method for Stochastic Optimization.” arXiv preprint
arXiv:1412.6980.
Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). “OpenNMT: Open-Source
Toolkit for Neural Machine Translation.” arXiv preprint arXiv:1701.02810.
Kuncoro, A., Dyer, C., Hale, J., Yogatama, D., Clark, S., and Blunsom, P. (2018). “LSTMs Can
Learn Syntax-Sensitive Dependencies Well, But Modeling Structure Makes Them Better.”
In Proceedings of ACL-2018 (1), pp. 1426–1436.
Lakew, S. M., Cettolo, M., and Federico, M. (2018). “A Comparison of Transformer and RNNs
on Multilingual NMT.” In Proceedings of Coling-2018, pp. 641–652.
Li, J., Xiong, D., Tu, Z., Zhu, M., Zhang, M., and Zhou, G. (2017). “Modeling Source Syntax
for Neural Machine Translation.” In Proceedings of ACL-2017 (1), pp. 688–697.
Luong, T., Pham, H., and Manning, C. D. (2015). “Effective Approaches to Attention-based
Neural Machine Translation.” In Proceedings of EMNLP-2015, pp. 1412–1421.
Ma, C., Liu, L., Tamura, A., Zhao, T., and Sumita, E. (2017). “Deterministic Attention for
Sequence-to-Sequence Constituent Parsing.” In Proceedings of AAAI-2017, pp. 3237–3243.
Ma, C., Tamura, A., Utiyama, M., Zhao, T., and Sumita, E. (2018). “Forest-based Neural Machine Translation.” In Proceedings of ACL-2018 (1), pp. 1253–1263.

Mi, H. and Huang, L. (2008). “Forest-based Translation Rule Extraction.” In Proceedings of EMNLP-2008, pp. 206–214.

Nakazawa, T., Yaguchi, M., Uchimoto, K., Utiyama, M., Sumita, E., Kurohashi, S., and Isahara, H. (2016). “ASPEC: Asian Scientific Paper Excerpt Corpus.” In Proceedings of LREC-2016, pp. 2204–2208.

Ng, N., Yee, K., Baevski, A., Ott, M., Auli, M., and Edunov, S. (2019). “Facebook FAIR’s WMT19 News Translation Task Submission.” In Proceedings of WMT-2019, pp. 314–319.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). “BLEU: a Method for Automatic Evaluation of Machine Translation.” In Proceedings of ACL-2002, pp. 311–318.

Post, M. (2018). “A Call for Clarity in Reporting BLEU Scores.” In Proceedings of WMT-2018, pp. 186–191.

Sennrich, R. and Haddow, B. (2016). “Linguistic Input Features Improve Neural machine translation.” In Proceedings of CMT-2016, pp. 83–91.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). “Self-Attention with Relative Position Representations.” In Proceedings of NAACL-2018 (2), pp. 464–468.

Strubell, E., Verga, P., Andor, D., Weiss, D., and McCallum, A. (2018). “Linguistically-Informed Self-Attention for Semantic Role Labeling.” In Proceedings of EMNLP-2018, pp. 5027–5038.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). “Sequence to Sequence Learning with Neural Networks.” In NIPS-2014, pp. 3104–3112.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). “Rethinking the Inception Architecture for Computer Vision.” In Proceedings of CVPR-2016, pp. 2818–2826.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). “Attention is All You Need.” In NIPS-2017, pp. 5998–6008.

Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2015). “Grammar as a Foreign Language.” In NIPS-2015, pp. 2773–2781.

Watanabe, T., Tamura, A., and Ninomiya, T. (2017). “CKY-based Convolutional Attention for Neural Machine Translation.” In Proceedings of IJCNLP-2017 (2), pp. 1–6.

Wu, S., Zhang, D., Yang, N., Li, M., and Zhou, M. (2017). “Sequence-to-Dependency Neural machine translation.” In Proceedings of ACL-2017 (1), pp. 698–707.

Wu, S., Zhang, D., Zhang, Z., Yang, N., Li, M., and Zhou, M. (2018). “Dependency-to-Dependency Neural Machine Translation.” IEEE/ACM TASLP, 26 (11), pp. 2132–2141.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao,
Ma et al. Syntax-based Transformer for Neural Machine Translation

Q., Macherey, K., et al. (2016). “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.” *arXiv preprint arXiv:1609.08144*

Zaremoodi, P. and Haffari, G. (2018). “Incorporating Syntactic Uncertainty in Neural Machine Translation with a Forest-to-Sequence Model.” In *Proceedings of Coling-2018*, pp. 1421–1429.

Chunpeng Ma: is currently a researcher in Fujitsu Laboratory. He received his B.E., M.E., Ph.D. from Harbin Institute of Technology in 2010, 2014 and 2020, respectively. His advisor was Prof. Tiejun Zhao. From July 2013 to January 2014, he interned at Baidu Inc. with instruction from Xianchao Wu. From October 2015 to September 2016, he interned at National Institute of Information and Communications Technology, with the instruction from Akihiro Tamura, cooperating with Lemao Liu. His research interests include syntactic parsing, machine translation and deep learning.

Akihiro Tamura: received his B.E., M.E., and Dr. Eng. from the Tokyo Institute of Technology in 2005, 2007, and 2013, respectively. After serving as a researcher at NEC and NICT, he worked as an assistant professor at Ehime University from 2017. Since 2020, he has been working as an associate professor at Doshisha University. His research interests include natural language processing and machine learning. He is a member of the Association for Computational Linguistics, the Information Processing Society of Japan, the Japanese Society for Artificial Intelligence, and the Association for Natural Language Processing.

Masao Utiyama: received the Ph.D. degree in engineering from the University of Tsukuba, Tsukuba, Japan, in 1997. He is currently a Research Manager in the National Institute of Information and Communications Technology, Tokyo, Japan. He is a member of the Association for Natural Language Processing, and Asia-Pacific Association for Machine Translation.

Eiichiro Sumita: received the M.S. degree in computer science from the University of Electro-Communications in 1982 and the Ph.D. degree in engineering from Kyoto University in 1999. He is the Associate Director General, ASTREC (Advanced Speech Translation Research and Development Promotion Center). He is currently the fellow of NICT. He is a member of the Association for Natural Language Processing, director of Japan Translation Federation, and
the president of Asia-Pacific Association for Machine Translation.

**Tiejun Zhao**: Doctor, professor of Research Center of Language Technology, School of Computer Science and Technology, Harbin Institute of Technology. He is associate dean of SCST and director of Ministry of Education-Microsoft Key Laboratory of NLP & Speech in HIT. He is associate director of Machine Translation Subject Committee of Chinese Information Processing Society, the member of editorial board of Journal of Chinese Information Processing, Journal of Automation, the senior member and associate director of Chinese Computing Committee of China Computer Federation. His research fields include: natural language understanding, content-based web information processing, applied artificial intelligence.

(Received November 26, 2019)
(Revised February 29, 2020)
(Accepted April 6, 2020)