Incorporating Source-Side Phrase Structures into Neural Machine Translation

Akiko Eriguchi*
Microsoft Research
akikoe@microsoft.com

Kazuma Hashimoto*
Salesforce Research
k.hashimoto@salesforce.com

Yoshimasa Tsuruoka
The University of Tokyo
Department of Information and
Communication Engineering
tsuruoka@logos.t.u-tokyo.ac.jp

Neural machine translation (NMT) has shown great success as a new alternative to the traditional Statistical Machine Translation model in multiple languages. Early NMT models are based on sequence-to-sequence learning that encodes a sequence of source words into a vector space and generates another sequence of target words from the vector. In those NMT models, sentences are simply treated as sequences of words without any internal structure. In this article, we focus on the role of the syntactic structure of source sentences and propose a novel end-to-end syntactic NMT model, which we call a tree-to-sequence NMT model, extending a sequence-to-sequence model with the source-side phrase structure. Our proposed model has an attention mechanism that enables the decoder to generate a translated word while softly aligning it with phrases as well as words of the source sentence. We have empirically compared the proposed model with sequence-to-sequence models in various settings on Chinese-to-Japanese and English-to-Japanese translation tasks. Our experimental results suggest that the use of syntactic structure can be beneficial when the training data set is small, but is not as effective as using a bi-directional encoder. As the size of training data set increases, the benefits of using a syntactic tree tends to diminish.

* The work was done when the first and second authors were at the University of Tokyo.

Submission received: 7 February 2018; revised version received: 11 February 2019; accepted for publication: 21 February 2019.

doi:10.1162/COLI_a_00348

1. Introduction

Machine translation has traditionally been one of the most complex language processing tasks, but recent advances of neural machine translation (NMT) make it possible to perform translation using a simple end-to-end architecture. In the Encoder-Decoder model (Cho et al. 2014b; Sutskever, Vinyals, and Le 2014), a recurrent neural network (RNN) called an *encoder* reads the whole sequence of source words to produce a fixed-length vector, and then another RNN called a *decoder* generates a sequence of target words from the vector. The Encoder-Decoder model has been extended with an *attention* mechanism (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015), which allows the model to jointly learn soft alignments between the source words and the target words. Recently, NMT models have achieved state-of-the-art results in a variety of language pairs (Wu et al. 2016; Zhou et al. 2016; Gehring et al. 2017; Vaswani et al. 2017).

In this work, we consider how to incorporate syntactic information into NMT. Figure 1 illustrates the phrase structure of an English sentence, which is represented as a binary tree. Each node of the tree corresponds to a grammatical phrase of the English sentence. Figure 1 also shows its translation in Japanese. The two languages are linguistically distant from each other in many respects; they have different syntactic constructions, and words and phrases are defined in different lexical units. In this example, the Japanese word "映画" is aligned with the English word "movie." The indefinite article "a" in English, however, is not explicitly translated into any Japanese words. One way to solve this mismatch problem is to consider the phrase structure of the English sentence and align the phrase "a movie" with the Japanese word "映画." The verb phrase of "went to see a movie last night" is also related to the eight-word sequence "昨晩 映画 を 見 に 行 っ た."

Since Yamada and Knight (2001) proposed the first syntax-based alignment model, various approaches to leveraging the syntactic structures have been adopted in statistical machine translation (SMT) models (Liu, Liu, and Lin 2006). In SMT, it is known that incorporating source-side syntactic constituents into the models improves word alignment (Yamada and Knight 2001) and translation accuracy (Liu, Liu, and Lin 2006; Neubig and Duh 2014). However, the aforementioned NMT models do not allow one to perform this kind of alignment.

To take advantage of syntactic information on the source side, we propose a syntactic NMT model. Following the phrase structure of a source sentence, we encode the

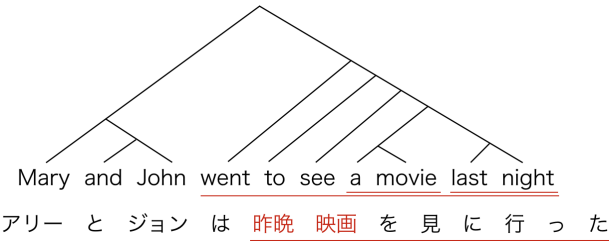

Figure 1
The phrase structure of the English sentence "Mary and John went to see a movie last night" and its Japanese translation.

sentence recursively in a bottom–up fashion to produce a sentence vector by using a tree-structured recursive neural network (RvNN) (Pollack 1990) as well as a sequential RNN (Elman 1990). We also introduce an attention mechanism to let the decoder generate each target word while aligning the input phrases and words with the output.

This article extends our conference paper on tree-to-sequence NMT (Eriguchi, Hashimoto, and Tsuruoka 2016) in two significant ways. In addition to an English-to-Japanese translation task, we have newly experimented with our tree-to-sequence NMT model in a Chinese-to-Japanese translation task, and observed that a bi-directional encoder was more effective than our tree-based encoder in both tasks. We also provide detailed analyses of our model and discuss the differences between the syntax-based and sequence-based NMT models. The article is structured as follows. We explain the basics of sequence-to-sequence NMT models in Section 2 and define our proposed tree-to-sequence NMT model in Section 3. After introducing the experimental design in Section 4, we first conduct experiments on two different tasks of {Chinese, English}-to-Japanese translation on a small scale and a series of analyses to understand the underlying key components in our proposed method in Section 5. Moreover, we report large-scale experimental results and analyses in the English-to-Japanese translation task in Section 6. In Section 7, we survey recent studies related to the syntax-based NMT models and conclude in Section 8 by summarizing the contributions of our work.

2. Sequence-to-Sequence Neural Machine Translation Model

2.1 Model Description

The sequence-to-sequence NMT models are built based on the idea of an *Encoder-Decoder* model, where an *encoder* converts each input sequence $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ into a vector space, and a *decoder* generates an output sequence $\boldsymbol{y} = (y_1, y_2, \cdots, y_m)$ from the vector, following the conditional probability of $P_\theta(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x})$. Here n is the input length, m is the output length, and θ denotes the model parameters. Figure 2 shows an illustration of the sequence-to-sequence NMT model.

The encoder computes the forward i-th hidden state $\overrightarrow{\boldsymbol{h}}^s_i \in \mathbb{R}^{d\times 1}$ by using an RNN as follows:

$$\overrightarrow{\boldsymbol{h}}^s_i = \overrightarrow{RNN}_{enc}\left(\overrightarrow{\boldsymbol{h}}^s_{i-1}, Emb(x_i)\right) \tag{1}$$

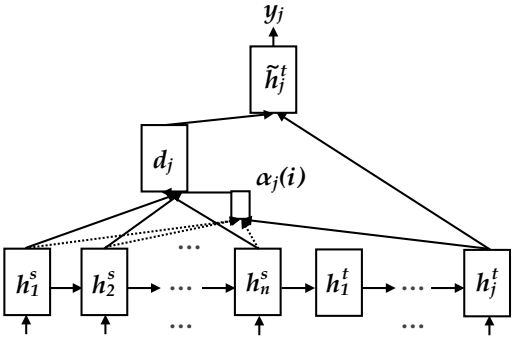

Figure 2
The overview of the sequence-based neural machine translation model.

where $\overrightarrow{\boldsymbol{h}}^{s}_{i-1} \in \mathbb{R}^{d\times 1}$ and $Emb(x_i) \in \mathbb{R}^{d\times 1}$ denote the previous hidden state and the word embedding of the i-th input word x_i, respectively. Applying the RNN function until the end of the source sequence, we obtain the last hidden vector $\overrightarrow{\boldsymbol{h}}^{s}_{n}$ representing a source sentence vector. In addition to the forward RNN units, we similarly compute backward RNN units as follows:

$$\overleftarrow{\boldsymbol{h}}^{s}_{i} = \overleftarrow{RNN}_{enc}\left(\overleftarrow{\boldsymbol{h}}^{s}_{i+1}, Emb(x_i)\right) \tag{2}$$

where we have another set of the backward RNN parameters. When using a bi-directional encoder in NMT models, the i-th source hidden state $\boldsymbol{H}_i \in \mathbb{R}^{2d\times 1}$ is computed as the concatenation of the forward and backward RNN units, which is denoted by $\boldsymbol{H}_i = [\overrightarrow{\boldsymbol{h}}^{s}_{i}; \overleftarrow{\boldsymbol{h}}^{s}_{i}]$. We consider the i-th forward hidden state $\overrightarrow{\boldsymbol{h}}^{s}_{i}$ as the i-th source hidden state $\boldsymbol{h}^{s}_{i}$, when the encoder is implemented as a forward RNN-based encoder.

The RNN units, RNN_{enc} in Equation (1), are often implemented with gated recurrent units (GRUs) (Cho et al. 2014b) or long short-term memory (LSTM) units (Hochreiter and Schmidhuber 1997; Gers, Schmidhuber, and Cummins 2000), and we use LSTM units in this article. Each LSTM unit contains four types of **gates** and two different types of hidden states, that is, a hidden unit $\boldsymbol{h}_t \in \mathbb{R}^{d\times 1}$ and a memory cell $\boldsymbol{c}_t \in \mathbb{R}^{d\times 1}$ at the time step t.

We also compute the j-th hidden state $\boldsymbol{h}^{t}_{j} \in \mathbb{R}^{d\times 1}$ $(j \geq 2)$ in the decoder by using another RNN as follows:

$$\boldsymbol{h}^{t}_{j} = RNN_{dec}\left(\boldsymbol{h}^{t}_{j-1}, Emb(y_{j-1})\right) \tag{3}$$

where $\boldsymbol{h}^{t}_{j-1} \in \mathbb{R}^{d\times 1}$ and $Emb(y_{j-1}) \in \mathbb{R}^{d\times 1}$ are the previous hidden state and the word embedding of the $(j-1)$-th output word y_{j-1}, respectively. We initialize the hidden state $\boldsymbol{h}^{t}_{1}$ and the cell unit $\boldsymbol{c}^{t}_{1}$ with the last hidden state and the cell unit from the encoder ($\boldsymbol{h}^{t}_{1} = \boldsymbol{h}^{s}_{n}$ and $\boldsymbol{c}^{t}_{1} = \boldsymbol{c}^{s}_{n}$).

When generating the outputs from the hidden states, the decoder is allowed to access the hidden states in the encoder and computes a **context vector** from the source hidden states and attention weights by the **attention mechanism** (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015). The context vector $\boldsymbol{d}_j \in \mathbb{R}^{d\times 1}$ at the time step j is computed as follows:

$$\boldsymbol{d}_j = \sum_{i=1}^{n} \boldsymbol{\alpha}_j(i)\boldsymbol{h}^{s}_{i} \tag{4}$$

$$\boldsymbol{\alpha}_j(i) = \frac{\exp(\boldsymbol{h}^{s}_{i} \cdot \boldsymbol{h}^{t}_{j})}{\sum_{k=1}^{n} \exp(\boldsymbol{h}^{s}_{k} \cdot \boldsymbol{h}^{t}_{j})} \tag{5}$$

where $\boldsymbol{h}^{s}_{i} \cdot \boldsymbol{h}^{t}_{j}$ is the inner product of $\boldsymbol{h}^{s}_{i}$ and $\boldsymbol{h}^{t}_{j}$ and is utilized as the similarity score between the vectors. $\boldsymbol{\alpha}_j(i)$ denotes the weight score of how relevant the i-th source state is to the word prediction at the j-th target state.

We prepare a new hidden state $\tilde{\boldsymbol{h}}_j^t \in \mathbb{R}^{d\times 1}$ to compute the j-th output conditional probability by using the softmax function as follows:

$$P(y_j|\boldsymbol{y}_{<j},\boldsymbol{x}) = softmax\left(\boldsymbol{W}_1\tilde{\boldsymbol{h}}_j^t + \boldsymbol{b}_1\right) \tag{6}$$

$$\tilde{\boldsymbol{h}}_j^t = \tanh\left(\boldsymbol{W}_2\left[\boldsymbol{h}_j^t;\boldsymbol{d}_j\right] + \boldsymbol{b}_2\right) \tag{7}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{|V^t|\times d}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d\times 2d}$, $\boldsymbol{b}_1 \in \mathbb{R}^{|V^t|\times 1}$, and $\boldsymbol{b}_2 \in \mathbb{R}^{d\times 1}$ are weight matrices and bias vectors, respectively, and $|V^t|$ stands for the target vocabulary size. $[\boldsymbol{h}_j^t;\boldsymbol{d}_j] \in \mathbb{R}^{2d\times 1}$ is the concatenation of $\boldsymbol{h}_j^t$ and $\boldsymbol{d}_j$. Introducing the **input-feeding** method proposed by Luong, Pham, and Manning (2015) to incorporate state $\tilde{\boldsymbol{h}}_j^t$ into the RNN states, we replace the decoder function in Equation (3) with the following equation:

$$\boldsymbol{h}_j^t = RNN_{dec}\left(\left[\boldsymbol{h}_{j-1}^t;\tilde{\boldsymbol{h}}_{j-1}^t\right], Emb(y_{j-1})\right) \tag{8}$$

where $[\boldsymbol{h}_j^t;\tilde{\boldsymbol{h}}_{j-1}^t] \in \mathbb{R}^{2d\times 1}$ is a concatenated vector. We also have confirmed the effectiveness of the input-feeding method in our preliminary experiments.

2.2 Training Model Parameters

We use a sentence-level parallel corpus $\mathcal{D}_{train} = \{(\boldsymbol{x}_1,\boldsymbol{y}_1), ..., (\boldsymbol{x}_N,\boldsymbol{y}_N)\}$ to train the NMT model parameters $\boldsymbol{\theta}$. The objective function of the NMT models is defined as the sum of the log-likelihood values of the translation pairs as follows:

$$J(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \log P(\boldsymbol{y}|\boldsymbol{x}) \tag{9}$$

$$= \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}}\sum_{j=1}^{m} \log P(y_j|\boldsymbol{y}_{<j},\boldsymbol{x}) \tag{10}$$

The model parameters $\boldsymbol{\theta}$ are learned through stochastic gradient descent (SGD).

3. Tree-to-Sequence Neural Machine Translation

The tree-to-sequence NMT model is a variant of the Encoder-Decoder model. On the source side, a tree-based encoder is constructed to represent the syntactic structure as well as sequential data. The tree-based encoder is a hybrid of the sequence-based encoder and a binary tree-based encoder modeled, respectively, with the RNN and an RvNN. Whereas the RNN-based encoder computes the uni-directional information on input sentences along the time series and discards the explicit syntactic information, the tree-based encoder directly leverages the syntactic structures of the sentences to compute phrase vectors. The overview of the tree-to-sequence NMT model is shown in Figure 3.

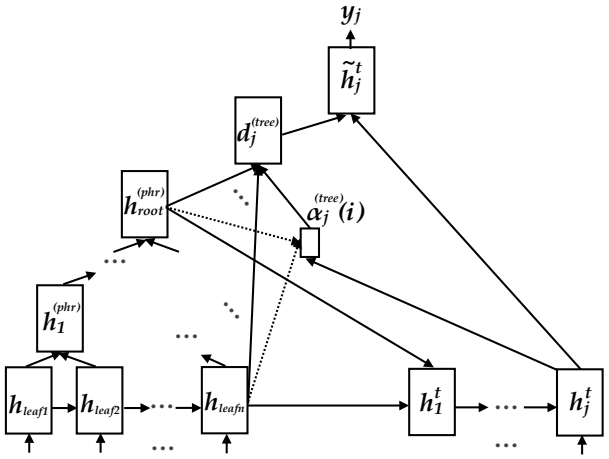

Figure 3
The overview of the tree-based neural machine translation model.

3.1 Tree-Based Encoder

The source sentence $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is parsed to obtain a binary parse tree by an external parser at a preprocessing step, as shown in Figure 1. Following the parse tree, the tree-based encoder calculates the k-th phrase hidden state $\boldsymbol{h}_k^{(phr)} \in \mathbb{R}^{d\times 1}$ by using an RvNN function as follows:

$$\boldsymbol{h}_k^{(phr)} = RvNN(\boldsymbol{h}_k^{left}, \boldsymbol{h}_k^{right}) \tag{11}$$

where $\boldsymbol{h}_k^{left} \in \mathbb{R}^{d\times 1}$ and $\boldsymbol{h}_k^{right} \in \mathbb{R}^{d\times 1}$ denote the hidden states of the left and right child nodes of the parent node $\boldsymbol{h}_k^{(phr)}$, respectively. As the i-th leaf node $\boldsymbol{h}_{leaf_i}$ corresponds to the i-th input word, the leaf nodes are initialized with the sequential units ($\boldsymbol{h}_{leaf_i} = \boldsymbol{h}_i^s$) computed by the sequence-based encoder. Starting from the leaf node vectors, we recursively use the RvNN function until the root node of the parse tree $\boldsymbol{h}_{root}^{(phr)}$ is computed. Consequently, we have two sentence vectors to represent the source-side input sentence: $\boldsymbol{h}_{root}^{(phr)}$ and $\boldsymbol{h}_n^s$.

We use the Tree-LSTM units proposed by Tai, Socher, and Manning (2015) in place of the *RvNN* units in Equation (11). The Tree-LSTM unit is a generalized LSTM unit that holds the k-th phrase hidden state $\boldsymbol{h}_k^{(phr)} \in \mathbb{R}^{d\times 1}$ for the k-th parent node computed as follows:

$$\boldsymbol{i}_k = \sigma(\boldsymbol{U}_l^{(i)}\boldsymbol{h}_k^{left} + \boldsymbol{U}_r^{(i)}\boldsymbol{h}_k^{right} + \boldsymbol{b}^{(i)}) \tag{12}$$

$$\boldsymbol{f}_k^l = \sigma(\boldsymbol{U}_l^{(f_l)}\boldsymbol{h}_k^{left} + \boldsymbol{U}_r^{(f_l)}\boldsymbol{h}_k^{right} + \boldsymbol{b}^{(f_l)}) \tag{13}$$

$$\boldsymbol{f}_k^r = \sigma(\boldsymbol{U}_l^{(f_r)}\boldsymbol{h}_k^{left} + \boldsymbol{U}_r^{(f_r)}\boldsymbol{h}_k^{right} + \boldsymbol{b}^{(f_r)}) \tag{14}$$

$$\boldsymbol{o}_k = \sigma(\boldsymbol{U}_l^{(o)}\boldsymbol{h}_k^{left} + \boldsymbol{U}_r^{(o)}\boldsymbol{h}_k^{right} + \boldsymbol{b}^{(o)}) \tag{15}$$

$$\tilde{\boldsymbol{c}}_k = \tanh(\boldsymbol{U}_l^{(\tilde{c})}\boldsymbol{h}_k^{left} + \boldsymbol{U}_r^{(\tilde{c})}\boldsymbol{h}_k^{right} + \boldsymbol{b}^{(\tilde{c})}) \tag{16}$$

$$\boldsymbol{c}_k^{(phr)} = \boldsymbol{i}_k \odot \tilde{\boldsymbol{c}}_k + \boldsymbol{f}_k^{left} \odot \boldsymbol{c}_k^{left} + \boldsymbol{f}_k^{right} \odot \boldsymbol{c}_k^{right} \tag{17}$$

$$\boldsymbol{h}_k^{(phr)} = \boldsymbol{o}_k \odot \tanh(\boldsymbol{c}_k^{(phr)}) \tag{18}$$

where $\boldsymbol{i}_k, \boldsymbol{f}_k^l, \boldsymbol{f}_k^r, \boldsymbol{o}_j, \tilde{\boldsymbol{c}}_j \in \mathbb{R}^{d\times 1}$ denote the input gate, the forget gates for the left and right child units, the output gate, and a state for updating the memory cell, respectively. $\boldsymbol{c}_k^{left}$ and $\boldsymbol{c}_k^{right}$ are the memory cells for the left and right child units, and $\boldsymbol{U}_{(\cdot)}^{(\cdot)} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}^{(\cdot)} \in \mathbb{R}^{d\times 1}$ represent a weight matrix and a bias vector, respectively. Unlike the conventional sequence-to-sequence models, because the tree-based encoder explicitly models the corresponding phrase nodes represented as a syntactic tree, our proposed model is expected to be more sensitive about not only an input sentence but also its structure, which may be helpful to resolve ambiguity in the sentence.

Our proposed tree-based encoder is a natural extension of the conventional sequential encoder because Tree-LSTM is a generalization of the chain-structured LSTM (Tai, Socher, and Manning 2015). Our encoder differs from the original Tree-LSTM in the calculation of the LSTM units for the leaf nodes, where we first compute word-level encoding with a (uni-directional) sequential encoder and then use the encodings as leaf nodes that are the inputs for the Tree-based encoder for phrase nodes. The motivation is to construct the phrase nodes in a context-sensitive way, which, for example, allows the model to compute different representations for multiple occurrences of the same word in a sentence because the sequential word-level encodings are computed in the context of the previous units. This ability contrasts with the original Tree-LSTM units, in which the leaves are composed only of the word embeddings without any contextual information.

3.2 Decoding Method with Tree-Based Encoder

Introducing the tree-based encoder to the sequence-based NMT model, we have two types of last hidden states $\boldsymbol{h}_n^s$ and $\boldsymbol{h}_{root}^{(phr)}$ computed from the sequence-based encoder and the tree-based encoder, respectively. The initial hidden state $\boldsymbol{h}_1^t$ of the decoder is computed by using another Tree-LSTM function formulated as Equations (12)–(18) and its parameters as follows:

$$\boldsymbol{h}_1^t = \mathit{TreeLSTM}\left(\boldsymbol{h}_n^s, \boldsymbol{h}_{root}^{(phr)}\right) \tag{19}$$

When a sentence fails to be parsed at the preprocessing step, we compute only the sequential hidden units and set $\boldsymbol{h}_{root}^{(phr)}$ to a zero vector ($\boldsymbol{h}_{root}^{(phr)} = \boldsymbol{0}$). The tree-based encoder leverages a parse tree if a sentence can be parsed; otherwise, the tree-based encoder works equally as the existing sequence-based encoder.

After obtaining the j-th hidden state $\boldsymbol{h}_j^t$ in the decoder following Equation (8), we introduce a new attention score $\boldsymbol{\alpha}_j^{(tree)}(i)$ for the tree-based encoder. Here, the j-th context

vector $\boldsymbol{d}_j^{(tree)}$ is calculated as follows:

$$\boldsymbol{d}_j^{(tree)} = \sum_{i=1}^{n} \boldsymbol{\alpha}_j^{(tree)}(i)\boldsymbol{h}_i^{(leaf)} + \sum_{i=n+1}^{2n-1} \boldsymbol{\alpha}_j^{(tree)}(i)\boldsymbol{h}_i^{(phr)} \tag{20}$$

$$\boldsymbol{\alpha}_j^{(tree)}(i) = \frac{\exp(\boldsymbol{h}_i \cdot \boldsymbol{h}_j^t)}{\sum_{l=1}^{2n-1} \exp(\boldsymbol{h}_l \cdot \boldsymbol{h}_j^t)} \tag{21}$$

where $\boldsymbol{h}_i$ denotes the i-th source hidden state, including the sequential hidden state $\boldsymbol{h}_i^s = \boldsymbol{h}_i^{(leaf)}$ $(i < n+1)$ and the i-th non-leaf phrase hidden state $\boldsymbol{h}_i^{(phr)}$ $(i > n)$. Note that a binarized tree with n leaf nodes has $(n-1)$ non-leaf nodes. This attention mechanism lets the decoder access both of the sequential hidden states and the syntactic hidden states of the hybrid encoder, and the tree-to-sequence model learns which type of units is highly weighted through training. Similar to the sequence-to-sequence NMT model introduced in Section 2, we compute the j-th output conditional probability, following Equations (6) and (7). Here, the context vector of $\boldsymbol{d}_j$ is replaced with $\boldsymbol{d}_j^{(tree)}$ computed by Equation (20).

3.3 Sampling-Based Approximation to the NMT Models

The computational cost in the softmax layer in Equation (6) occupies most of the training time because the cost increases linearly with the size of the vocabulary. A variety of approaches addressing this problem have been proposed, including negative sampling methods such as BlackOut sampling (Ji et al. 2016) and noise-contrastive estimation (NCE) (Gutmann and Hyvärinen 2012), and binary code prediction (Oda et al. 2017). BlackOut has been shown to be effective in training RNN language models even with one-million-word vocabulary on CPUs.

The NMT models are trained as RNN-based conditional language models in Equation (10), and we apply the BlackOut sampling technique to the NMT models. We redefine the objective function as follows:

$$J(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \sum_{j=1}^{|\boldsymbol{y}|} \left(\log \tilde{p}(y_j \mid \boldsymbol{y}_{<j}, \boldsymbol{x}) + \sum_{k\in S^K} \left(1 - \log \tilde{p}(y^k \mid \boldsymbol{y}_{<j}, \boldsymbol{x})\right) \right) \tag{22}$$

$$\tilde{p}(y_j \mid \boldsymbol{y}_{<j}, \boldsymbol{x}) = \frac{q_j \exp(s_j^j)}{q_j \exp(s_j^j) + \sum_{k\in S^K} \left(q_k \exp(s_j^k)\right)} \tag{23}$$

where $|\boldsymbol{y}|$ denotes the length of each target sentence $\boldsymbol{y}$, and y^k and S^K denote a negative sample word and a smaller vocabulary that holds K negative samples drawn from the original large vocabulary ($|V^t|$) by using a sampling distribution ($Q(y)$). The j-th output word y_j is not included in S^K. q_j is computed as $q_j \propto \frac{1}{Q(y_j)}$. s_j^k represents the score for the k-th word in the vocabulary $|V^t|$ at the j-th step. At each word prediction step in the training, BlackOut estimates the conditional probability in Equation (6) for the target

word and K negative samples using a weighted softmax function. $\boldsymbol{s}_j^k$ is computed as follows:

$$s_j^k = \boldsymbol{w}_k \tilde{\boldsymbol{h}}_j^t + b_k \tag{24}$$

where $\boldsymbol{w}_k$ and b_k are the k-th row of $\boldsymbol{W}_1$ and the k-th element of $\boldsymbol{b}_1$. The negative samples are drawn from the unigram distribution raised to the power $\beta \in [0, 1]$ (Mikolov et al. 2013):

$$Q(w) \propto p_{unigram}^{\beta}(y) \tag{25}$$

The unigram distribution $p_{unigram}^{\beta}(y)$ is estimated on the training data and β is a hyperparameter. BlackOut is closely related to NCE (Gutmann and Hyvärinen 2012) and achieves better perplexity scores than the original softmax and NCE in RNN language models. The advantages of Blackout over the other methods are discussed in Ji et al. (2016). When using BlackOut, the original full softmax formula can be applied at test time, and there are no additional model parameters for BlackOut.

4. Experimental Design

4.1 Experimental Settings

Chinese-to-Japanese. We used the Asian Scientific Paper Excerpt Corpus (ASPEC) (Nakazawa et al. 2016) for the Chinese-to-Japanese translation provided by the Workshop of Asian Translation 2015 (WAT2015). Following the official preprocessing steps,[1] we tokenized the data by using *KyTea* (Neubig, Nakata, and Mori 2011) as a Japanese segmenter and *Stanford Segmenter* as a Chinese word segmenter.[2] We discarded a sentence pair when either of the sentences had more than 50 words. We used the *Stanford Parser* (Levy and Manning 2003)[3] as an external parser and parsed the Chinese sentences. We did not use any more specific information output by the parsers such as phrase labels. If a sentence fails to be parsed, it is re-parsed by a simpler probabilistic context-free grammar parser. We used a script[4] to convert the parse trees to their corresponding binary trees because our tree-to-sequence model assumes binary trees as inputs. We used the first 100,000 parallel sentences from the training data to investigate the effectiveness of the proposed NMT model. The vocabularies were composed of the words appearing in the training data more than or equal to N times. We set $N = 2$ for Japanese and $N = 3$ for Chinese. The out of vocabulary words were mapped to the special token "unk," and we inserted another special symbol "EOS" at the end of all the sentences. The vocabulary sizes of Chinese and Japanese are 29,011 and 32,640, respectively. Table 1 shows the statistics on the data set of the Chinese-to-Japanese translation task.

1 `http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/automatic_evaluation_systems/tools.html`, `http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/automatic_evaluation_systems/automaticEvaluationZH.html`.

2 We used a FULL SVM model of KyTea (jp-0.4.2-utf8-1.mod) and a Chinese Penn Treebank model of Stanford Segmentor version 2014-06-16 as suggested in WAT 2015.

3 The Stanford Parser version is 2017-06-09 with a Chinese-Factored model.

4 We used *TreeBinarizer.java* available in `https://github.com/stanfordnlp/CoreNLP`.

Table 1
Statistics on the Chinese-to-Japanese data set in the ASPEC corpus. The number of "Sentences" shows the sum of the "Parsed (Chinese-Factored)" sentences and the "Parsed (PCFG)" sentences.

	Sentences	Parsed (Chinese-Factored)	Parsed (PCFG)
Train	100,000	95,647	4,353
Development	2,090	1,779	311

Table 2
Statistics on the English-to-Japanese data set in ASPEC corpus. The number of "Sentences" is equal to the total number of the "Parsed successfully" sentences and the "Parsed unsuccessfully" sentences.

	Sentences	Parsed successfully	Parsed unsuccessfully
Large Train	1,353,635	1,346,946	6,689
Small Train	100,000	99,541	459
Development	1,790	1,779	11
Test	1,812	1,801	11

English-to-Japanese. We used the ASPEC corpus (Nakazawa et al. 2016) for the English-to-Japanese translation task provided by WAT2015. We used *Enju* (Miyao and Tsujii 2008), a head-driven phrase structure grammar (Sag, Wasow, and Bender 2003) parser for English, and the tokenization in English follows the Enju parser.[5] The English corpus was lowercased. We followed the official preprocessing for the Japanese corpus as in the Chinese-to-Japanese experimental settings. We used Enju only to obtain a binary phrase structure for each source-side sentence. We removed the sentence pairs in which either of the sentences is longer than 50 words. Enju returns either *success* or *failure* after parsing an English sentence. When the Enju parser fails to parse a sentence, it is treated as a sequence of words in the proposed model.[6] Table 2 shows the statistics on the data set of English-to-Japanese translation task.

We build a small training corpus by extracting the first 100,000 parallel sentences from the training data in the English-to-Japanese translation task. The vocabularies were constructed with the words appearing in the training data no less than N times as well as in the Chinese-to-Japanese translation task. We set $N = 2$ and $N = 5$ for the small and large training data set, respectively. The out of vocabulary words were mapped to "unk," and "EOS" was inserted at the end of each sentence. The vocabulary sizes of English and Japanese are (87,796; 65,680) and (25,456; 23,509) in the large and small training data set, respectively. When training the models on the large data set, we follow the same parameter setting except that our proposed model has 512-dimensional word embeddings and d-dimensional hidden units ($d \in \{512, 768, 1{,}024\}$). K is set to 2,500. When utilizing beam search to generate a target sentence for a source sentence at test time, we selected the optimal beam width found on the development data set.

5 Contrary to Stanford Parser, Enju returns a binarized tree.

6 When the Enju parser fails to parse a sentence because of "sentence length limit exceeded," we let the sentence be parsed again with an additional option of "-W 200" to increase the limit size of sentences up to 200. We found one sentence in both the development data and test data that is parsed again with the option.

We evaluated the models by two automatic evaluation metrics, RIBES (Isozaki et al. 2010) and BLEU (Papineni et al. 2002) following WAT 2015. We used the KyTea-based evaluation script for the translation results.[7] The RIBES score is a metric based on rank correlation coefficients with word precision, and this score is known to have stronger correlation with human judgments than BLEU in translation between English and Japanese, as discussed in Isozaki et al. (2010). The BLEU score is based on n-gram word precision and a brevity penalty for outputs shorter than the references.

4.2 Training Details

We conduct experiments with the sequence-to-sequence NMT model as a baseline and our proposed model described in Sections 2 and 3, respectively. Each model has 256-dimensional hidden units and word embeddings. The biases, softmax weights, and BlackOut weights are initialized with zeros. The hyperparameter β of BlackOut is set to 0.4 as recommended by Ji et al. (2016). The number of negative samples K in BlackOut was set to $K \in \{500, 2000\}$. Here, we shared the negative samples of each target word in a sentence in training time, following Hashimoto and Tsuruoka (2017).

Following Józefowicz, Zaremba, and Sutskever (2015), we initialize the forget gate biases of LSTM and Tree-LSTM with 1.0. The remaining model parameters in the NMT models in our experiments are uniformly initialized in $[-0.1, 0.1]$. The model parameters are optimized by plain SGD with the mini-batch size of 128. The initial learning rate of SGD is 1.0. When the development loss becomes worse per epoch, we halve the learning rate from the next epoch until it converges. When INF/NAN values appear in a mini-batch during training, we skip the mini-batch training. Gradient norms are clipped to 3.0 to avoid exploding gradient problems (Pascanu, Mikolov, and Bengio 2012).

4.3 Beam Search with Penalized Length

We use beam search to decode a target sentence for an input sentence $\boldsymbol{x}$ and calculate the sum of the log-likelihood values of the target sentence $\boldsymbol{y} = (y_1, \cdots, y_m)$ as the beam score:

$$score(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{m} \log p(y_j | \boldsymbol{y}_{<j}, \boldsymbol{x}) \tag{26}$$

Decoding in the NMT models is a generative process and depends on the target language model given a source sentence. The score becomes smaller as the target sentence becomes longer, and thus the simple beam search does not work well when decoding a long sentence, as reported in Cho et al. (2014a), Pouget-Abadie et al. (2014), and Koehn and Knowles (2017).

We apply a method to utilize statistics on sentence lengths in beam search. Assuming that the length of a target sentence correlates with the length of the source sentence,

7 http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/automatic_evaluation_systems/automaticEvaluationJA.html.

we redefine the score of each candidate as follows:

$$score(\boldsymbol{x}, \boldsymbol{y}) = Length_{\boldsymbol{x},\boldsymbol{y}} + \sum_{j=1}^{m} \log p(y_j | \boldsymbol{y}_{<j}, \boldsymbol{x}) \tag{27}$$

$$Length_{\boldsymbol{x},\boldsymbol{y}} = \log p(length(\boldsymbol{y}) | length(\boldsymbol{x})) \tag{28}$$

where $Length_{\boldsymbol{x},\boldsymbol{y}}$ denotes the penalty for the conditional probability of the target sentence length $length(\boldsymbol{y})$ given the source sentence length $length(\boldsymbol{x})$. Once the EOS token is predicted in a top candidate during beam search, the length penalty score is added to it. The beam search with the length penalty score lets the model generate a sentence while considering the expected length of the target sentence given a source sentence whose length is $length(\boldsymbol{x})$ on the statistics. In our experiments, we computed the conditional probability $p(length(\boldsymbol{y})|length(\boldsymbol{x}))$ in advance, following the statistics collected in the first 1.5 million sentences of the English-to-Japanese training data set and all the 0.7 million sentences of the Chinese-to-Japanese training data set. We allow the decoder to generate up to 300 and 150 words in the Chinese-to-Japanese and English-to-Japanese translation tasks, respectively.

We used the beam search to generate a target sentence for a source sentence at test time. We set the beam width to 20, unless otherwise stated.

5. Experimental Results and Discussion on Small Data Sets

In this section, we first report the experimental results on small data sets for Chinese-to-Japanese and English-to-Japanese translation tasks. To understand the effectiveness of our proposed methods, we conduct a series of analyses on the proposed beam search, sequential LSTM units for the tree-based encoder, and the model capacity.

5.1 Experimental Results on Small Data Sets

Tables 3 and 4 summarize the experimental results of translation accuracy in the Chinese-to-Japanese and English-to-Japanese translation tasks, respectively. Each table reports the values of perplexity, RIBES, BLEU, and the training time on the development data with the NMT models. We conducted the experiments with our proposed methods and the baseline of the sequence-to-sequence model using BlackOut and the original softmax. We generated each translation by our proposed beam search with a beam size of 20. In both tables, we report the experimental results obtained by feeding the reversed inputs into the sequence-to-sequence NMT models (shown as "w/ reverse inputs" in tables), and by utilizing the bi-directional encoders. We ran the bootstrap re-sampling method (Koehn 2004) and observed a statistical significant difference on both RIBES and BLEU scores between the proposed method and the sequence-to-sequence NMT model with reversed inputs in all the settings. The symbol † indicates that our proposed model significantly outperforms the sequence-to-sequence NMT model both without and with reversed inputs except the model "w/ bi-directional encoder" in the corresponding settings on K ($p < 0.05$).

As the negative sample size K increases, we can see that both the proposed model (Tree-to-Sequence NMT) and the sequence-to-sequence NMT models (Sequence-to-Sequence NMT, Sequence-to-Sequence NMT w/ reverse inputs, and Sequence-to-Sequence NMT w/ bi-directional encoder) improve the translation accuracy in both

Table 3
Evaluation results on the development data in the Chinese-to-Japanese translation task. *K* denotes the size of negative samples in the BlackOut sampling method. Here, we implemented the sequence-to-sequence NMT model (Luong, Pham, and Manning 2015).

	K	*Perp*.	RIBES	BLEU
Tree-to-Sequence NMT	500	13.4	76.7	25.5
	2,000	13.3	77.8	26.9
Tree-to-Sequence (*softmax*)	—	11.3	78.0	27.0
Sequence-to-Sequence NMT	500	13.2	76.9	25.9
w/ reverse inputs	500	14.7	75.8	25.0
w/ bi-directional encoder	500	12.7	77.5	27.1
Sequence-to-Sequence NMT	2,000	12.8	77.4	26.7
w/ reverse inputs	2,000	15.7	77.1	26.2
w/ bi-directional encoder	2,000	13.8	78.6	27.8
Sequence-to-Sequence NMT (*softmax*)	—	11.8	78.0	26.8
w/ reverse inputs	—	13.8	77.1	26.2
w/ bi-directional encoder	—	11.7	78.8	28.3

Table 4
Evaluation results on the development data using the small training data in the English-to-Japanese translation task. The training time per epoch is also shown, and *K* is the number of negative samples in BlackOut. Here, we implemented the sequence-to-sequence NMT model (Luong, Pham, and Manning 2015). The last column "Time" reports the training time (min./epoch).

	K	*Perp*.	RIBES	BLEU	Time
Tree-to-Sequence NMT	500	16.2	75.8†	24.8†	7.0
	2,000	16.9	75.9	24.9	8.9
Tree-to-Sequence NMT (*softmax*)	—	15.4	76.4	25.6†	117.2
Sequence-to-Sequence NMT	500	16.9	75.3	24.0	5.4
w/ reverse inputs	500	18.8	74.7	23.0	
w/ bi-directional encoder	500	15.5	76.5	25.9	7.8
Sequence-to-Sequence NMT	2,000	19.0	76.0	24.7	7.3
w/ reverse inputs	2,000	21.3	75.4	23.9	
w/ bi-directional encoder	2,000	15.9	77.1	26.9	9.4
Sequence-to-Sequence NMT (*softmax*)	—	16.1	76.2	24.9	112.1
w/ reverse inputs	—	17.2	75.3	24.0	
w/ bi-directional encoder	—	14.9	76.8	26.8	130.2

of the evaluation metrics. The gains in the Chinese-to-Japanese translation task are +1.1 RIBES and +1.4 BLEU for the proposed model, and +0.5 RIBES, +0.8 BLEU for the sequence-to-sequence model, +1.3 RIBES and +1.2 BLEU for the reversed inputs model, and +1.1 RIBES and +0.7 BLEU for the bi-directional NMT model; we also observe similar trends in the English-to-Japanese translation task. Because the BlackOut sampling is an approximation method for softmax, the models are expected to obtain better accuracy when the negative examples (K) increase. Compared to the results obtained by the models using the softmax layer, the tree-to-sequence NMT model with $K = 2{,}000$ achieves a competitive score in the Chinese-to-Japanese translation task. We also found that better perplexity does not always lead to better translation scores with BlackOut, as shown in Table 3. One of the possible reasons is that BlackOut distorts

the target word distribution by the modified unigram-based negative sampling where frequent words can be treated as negative samples multiple times at each training step.

As to the results of the sequence-to-sequence NMT models, reversing the word order in the input sentence decreases the scores in Chinese-to-Japanese and English-to-Japanese translation, which contrasts with the results of other language pairs reported in previous work (Sutskever, Vinyals, and Le 2014; Luong, Pham, and Manning 2015).

The rightmost column of Time in Table 4 reports the training time (min.) per epoch of each NMT model. Although the result of softmax is better than those of BlackOut ($K = 500, 2{,}000$), the training time of softmax per epoch is about 13 times longer than that of BlackOut even with the small data set.[8] By taking syntactic information into consideration, our proposed model improves the scores, outperforming the sequence-based NMT models with reverse inputs in both tasks. The gains of BLEU are larger in the English-to-Japanese translation task than in the Chinese-to-English task. However, extending the sequential encoder to the bi-directional one improves both evaluation scores by taking a slightly additional time for training.

5.2 Effects of Beam Search with Length Penalty

Table 5 shows the results on the development data of our proposed method with BlackOut ($K = 2{,}000$) by the simple beam search and our proposed beam search. The beam size is set to $\{10, 15, 20\}$ in the simple beam search, and to 20 in our proposed search. The brevity penalty value in BLEU denotes the ratio of the hypothesis length over the reference length. We can see that our proposed search outperforms the simple beam search in BLEU score without losing the score of RIBES. Unlike RIBES, the BLEU score is sensitive to the beam size and becomes lower as the beam size increases. We found that the brevity penalty had a relatively small impact on the BLEU score in the simple beam search as the beam size increased. Our search method works better than the simple beam search by keeping long sentences in the candidates with a large beam size.

5.3 Discussion on Small Data Sets

Here, we conduct detailed analyses to explore what kind of component is the key in the proposed methods.

Effects of the Sequential LSTM Units. We investigated the effects of the sequential LSTM units at the leaf nodes in our proposed tree-based encoder. Table 6 shows the result on the development data of our proposed encoder and that of an attention-based tree-based encoder without sequential LSTM units. For this evaluation, we used the 1,779 sentences that were successfully parsed by Enju because the encoder without sequential LSTM units always requires a parse tree. The results show that our proposed encoder considerably outperforms the encoder without sequential LSTM units, suggesting that the sequential LSTM units at the leaf nodes contribute to the context-aware construction of the phrase representations in the tree.

8 We applied Qiao et al. (2017)'s approach, which effectively uses the cache to speed up the computations of matrices on CPUs.

Table 5
Effects of the beam search on the Chinese-to-Japanese and English-to-Japanese development data.

	Beam size	BLEU (brevity penalty)	RIBES
Chinese-to-Japanese			
Proposed beam search	20	**27.0** (0.92)	**78.0**
Standard beam search	10	25.4 (0.85)	77.4
	15	25.1 (0.84)	77.9
	20	24.9 (0.83)	77.8
English-to-Japanese			
Proposed beam search	20	**25.6** (0.93)	76.4
Standard beam search	10	24.8 (0.89)	76.2
	15	24.8 (0.88)	76.4
	20	24.6 (0.87)	76.4

Table 6
Effects of the sequential LSTM units in our proposed tree-based encoder on the development data, including parsed data, in the Chinese-to-Japanese and English-to-Japanese translation tasks.

	RIBES	BLEU
Chinese-to-Japanese		
with sequential LSTM units	**78.0**	**27.0**
without sequential LSTM units	71.0	23.0
English-to-Japanese		
with sequential LSTM units	**76.4**	**25.7**
without sequential LSTM units	71.7	20.8

Effects of Model Capacity. Here, we conduct a control experiment of model parameters to investigate whether the proposed tree-to-sequence NMT model has the benefit of more parameters than the sequence-to-sequence NMT model does. We set the hidden and embedding dimension sizes of the sequence-to-sequence NMT model to $d = 269$ and $d = 272$ in the Chinese-to-Japanese and English-to-Japanese translation tasks, respectively, to make the number of parameters of the model equal to that of the tree-to-sequence NMT model. The numbers of parameters of the models are 25.6M and 21.4M in the setup. Table 7 reports the results on both evaluation scores, and we used the proposed beam search method with a width of 20. As a result in the English-to-Japanese translation task, the proposed model obtained better RIBES and BLEU scores than the sequence-to-sequence NMT model both with and without reversed input with the same number of parameters. On the other hand, the same significant improvement is observed only over the sequence-to-sequence NMT model with reversed inputs in the Chinese-to-Japanese translation task.

Comparison with Dependency-to-Sequence NMT models. Here, we have the comparison of the performance with a syntax-based NMT model, which incorporates source dependency trees into the encoder. We used the dependency-to-sequence NMT architecture

Table 7
Comparison of the MT performance between the sequence-to-sequence NMT models and the tree-to-sequence NMT models with the same model sizes. We mark a score with † and ‡ to show the significant improvement over the Sequence-to-Sequence NMT model with reverse inputs and both Sequence-to-Sequence NMT models ($p < 0.05$), respectively. A statistical significance test was performed by using the bootstrap re-sampling method (Koehn 2004).

	RIBES	BLEU
Chinese-to-Japanese		
Tree-to-Sequence NMT	78.0†	27.0†
Sequence-to-Sequence NMT	77.8	26.7
w/ reverse inputs	77.2	26.2
English-to-Japanese		
Tree-to-Sequence NMT	76.4‡	25.6‡
Sequence-to-Sequence NMT	75.8	24.9
w/ reverse inputs	75.8	24.6

proposed by Hashimoto and Tsuruoka (2017), where the model architecture is considered as a one-layered graph convolutional neural network and the source input texts with its predicted dependency are used to generate a translation. Following the setup of Hashimoto and Tsuruoka (2017), we first pretrained the dependency-to-sequence NMT model in a dependency parsing task and then trained it in a machine translation task. We used the same experimental settings and training procedures described in Section 4. Table 8 reports the performance of a dependency-to-sequence NMT model and the proposed tree-to-sequence NMT model. We let the decoder output the translation on the development data set by using the beam search with penalty length with a beam width of 20. We confirmed that the proposed tree-to-sequence NMT model outperforms the dependency-to-sequence NMT model on RIBES and BLEU scores in both translation tasks.

Table 8
Comparison with other syntax-based approaches which leverages source syntax structures.

	RIBES	BLEU
Chinese-to-Japanese		
Proposed: Tree-to-Sequence NMT	77.8	26.9
Dependency-to-Sequence NMT reimplementation of Hashimoto and Tsuruoka (2017)	77.5	25.5
English-to-Japanese		
Proposed: Tree-to-Sequence NMT	76.0	23.9
Dependency-to-Sequence NMT reimplementation of Hashimoto and Tsuruoka (2017)	74.9	22.9

Table 9
Evaluation results on the test data in the English-to-Japanese translation task.

NMT Model	RIBES	BLEU
Tree-to-Sequence NMT model ($d = 512$)	81.66	35.05
Tree-to-Sequence NMT model ($d = 768$)	81.83	35.73
Tree-to-Sequence NMT model ($d = 1{,}024$)	81.94	35.57
Ensemble of the above three models	**83.27**	38.00
Sequence-to-Sequence NMT model ($d = 512$) reimplementation of Luong, Pham, and Manning (2015)	81.60	34.64
Seq-to-Seq NMT with bi-directional LSTM encoder (Zhu 2015)	79.70	32.19
+ Ensemble, *unk* replacement	80.27	34.19
+ System combination, 3 pre-reordered ensembles	80.91	36.21
Seq-to-Seq NMT with bi-directional GRU encoder (Lee et al. 2015) + character-based decoding, Begin/Inside representation	81.15	35.75
SMT Model		
Baseline: Phrase-based SMT model	69.19	29.80
Baseline: Hierarchical Phrase-based SMT model	74.70	32.56
Baseline: Tree-to-String SMT model	75.80	33.44
Tree-to-String SMT model (Neubig and Duh 2014)	79.65	36.58
+ Seq-to-Seq NMT Rerank (Neubig, Morishita, and Nakamura 2015)	81.38	38.17

6. Experimental Results and Discussion on Large Data Set

6.1 Experimental Results on Large Data Set

Table 9 shows the experimental results of RIBES and BLEU scores achieved by the trained models on the large data set.[9] The results of the other systems are the ones reported in Nakazawa et al. (2015). All of our proposed models show similar performance regardless of the value of d. Our ensemble model is composed of the three models with $d = 512$,768, and 1,024, and it shows the best RIBES score among all systems. As for the time required for training, our implementation needs about 8 hours to perform one epoch on the large training data set with $d = 512$. It would take about 8 days without using the BlackOut sampling.[10]

Comparison with the NMT Models. Compared with our reimplementation of the sequence-to-sequence NMT model (Luong, Pham, and Manning 2015), we did not observe a significant difference from the tree-to-sequence model ($d = 512$). The model of Zhu (2015) is an NMT model (Bahdanau, Cho, and Bengio 2015) with a bi-directional LSTM encoder, and uses 1,024-dimensional hidden units and 1,000-dimensional word embeddings. The model of Lee et al. (2015) is also an NMT model with a bi-directional GRU encoder, and uses 1,000-dimensional hidden units and 200-dimensional word embeddings. Both models are sequence-based NMT models. Our single proposed model with $d = 512$ outperforms Zhu (2015)'s sequence-to-sequence NMT model with a bi-directional LSTM encoder by +1.96 RIBES and by +2.86 BLEU scores.

9 We found two sentences that end without *EOS* with $d = 512$, and then we decoded them again with the beam size of 1,000 following (Zhu 2015).

10 We run the experiments on multi-core CPUs; 16 threads on Intel Xeon CPU E5-2667 v3 @ 3.20 GHz.

Comparison with the SMT models. The baseline SMT systems are phrase-based, hierarchical phrase-based, and tree-to-string systems in WAT 2015 (Nakazawa et al. 2015), and the system submitted by Neubig, Morishita, and Nakamura (2015) achieves the best performance among the tree-to-string SMT models. Each of our proposed models outperforms the SMT models in RIBES and achieved almost as high performance as Neubig, Morishita, and Nakamura (2015)'s system in BLEU.

6.2 Qualitative Analysis in English-to-Japanese Translation

We illustrate the translations of test data by our model with $d = 512$ and several attention relations when decoding a sentence. In Figures 4 and 5, an English sentence represented as a binary tree is translated into Japanese, and several attention relations between English words or phrases and Japanese words are shown with the highest attention score α. The additional attention relations are also illustrated for comparison. We see that the target words softly aligned with source words and phrases.

In Figure 4, the Japanese word "液晶" means "liquid crystal," and it has a high attention score ($\alpha = 0.41$) with the English phrase "liquid crystal for active matrix." This is because the j-th target hidden unit $\boldsymbol{s}_j$ has the contextual information about the previous words $\boldsymbol{y}_{<j}$ including "活性 マトリックス の" ("for active matrix" in English). The Japanese word "セル" is softly aligned with the phrase "the cells" with the highest attention score ($\alpha = 0.35$). In Japanese, there is no definite article like "the" in English, and it is usually aligned with *null*, as described in Section 1.

In Figure 5, in the case of the Japanese word "示" ("showed" in English), the attention score with the English phrase "showed excellent performance" ($\alpha = 0.25$) is higher than that with the English word "showed" ($\alpha = 0.01$). The Japanese word "の" ("of" in English) is softly aligned with the phrase "of Si dot MOS capacitor" with the highest attention score ($\alpha = 0.30$). It is because our attention mechanism takes each word of the previous context of the Japanese phrases "優れた性能" ("excellent performance" in English) and "Ｓｉ ドット ＭＯＳ コンデンサ" ("Si dot MOS capacitor" in English) into account and softly aligned the target words with the whole phrase when translating the English verb "showed" and the preposition "of." Our proposed model can thus flexibly learn the attention relations between English and Japanese.

We observed that our model translated the word "active" into "活性," a synonym of the reference word "アクティブ." We also found similar examples in other sentences, where our model outputs synonyms of the reference words, for instance, "女 and "女性" ("female" in English) and "NASA" and "航空宇宙局" ("National Aeronautics and Space Administration" in English). These translations are penalized in terms of BLEU scores,

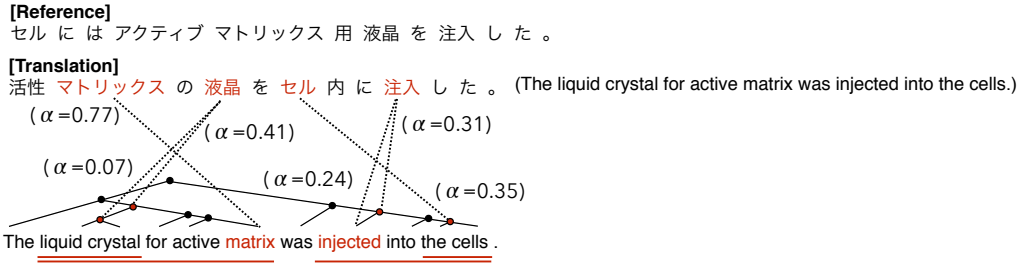

Figure 4
A translation example of a short sentence and the attention relations by our proposed model.

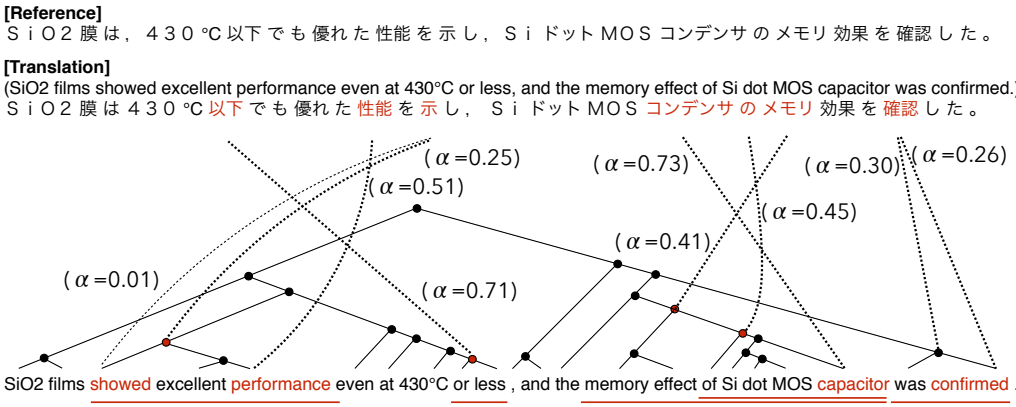

Figure 5
A translation example of a long sentence and the attention relations by our proposed model.

but they do not necessarily mean that the translations were wrong. This point may be supported by the fact that the NMT models were highly evaluated in WAT 2015 by crowd sourcing (Nakazawa et al. 2015).

6.3 Error Analyses on Sequence-to-Sequence and Tree-to-Sequence NMT Models

To investigate what kinds of errors are harmful in the NMT models, we took 100 examples randomly from the development data in the English-to-Japanese translation task and classified the error types found in the generated translations obtained by the sequence-to-sequence NMT model and tree-to-sequence NMT models described in Section 6. In Table 10, we can see that both NMT models suffered from the word-level undertranslation and wrong translation issues with the highest error ratio. As for the phrase-level undertranslation and wrong translation, the tree-to-sequence NMT model successfully decreased the number of the errors. However, the sequence-to-sequence NMT model has fewer errors of repetition than the tree-to-sequence NMT model. The reason for this can be that the phrase-based attention mechanism enhances the decoder to generate a translation for specific phrases. The phrase-based repetitions are not trained through teacher forcing training, where the models are allowed to because optimized on a word-scale translation rather than a phrase-scale.

Table 10
Error types of sequence-to-sequence NMT model (Seq-to-Seq NMT) and the proposed tree-to-sequence NMT model (Tree-to-Seq NMT) on development data and each error ratio (%). Note that the different types of errors can be found in the same translations.

	Seq-to-Seq NMT (%)	Tree-to-Seq NMT (%)
Undertranslation (word)	29 (38.2%)	29 (42.6%)
Undertranslation (phrase)	16 (21.1%)	8 (11.8%)
Wrong translation (word)	22 (28.9%)	17 (25.0%)
Wrong translation (phrase)	2 (2.6%)	0 (0.0%)
Repetition	7 (9.2%)	14 (20.6%)

Table 11
Translation error examples generated by the sequence-to-sequence NMT model and the translation changes when manually fixing source sentences.

Original source sentence: First, after static electricity of human body is released a Eathernet card is fixed. Reference: 先ず人体の静電気を逃がしてから，イーサネットカードを装着する。 Translation: まず，イーサネットカードで人体の静電気を解放している (First, static electricity of human body is released by an Ethernet card.)
Manually Revised Source Sentence A: First, after static electricity of human body is released, a Ethernet card is fixed. Translation A: 最初に，人間の静電気をイーサネットカードであることを前提にしている。 (At first, static electricity of human is assumed to be an Eathernet card.)
Manually Revised Source Sentence B: First, a Ethernet card is fixed after static electricity of human is released. Translation B: 最初に，人間の静的電気後にイーサネットカードを固定している。 (At first, an Eathernet card is set after static electricity of human body.)
Manually Revised Source Sentence C: First, a Ethernet card is fixed, after static electricity of human body is released. Translation C: 最初に，イーサネットカードは人体の静電気を解放している。 (At first, an Eathernet card releases static electricity of human body.)

In Table 11, we show some examples of the phrase-level undertranslation and wrong translation outputs by the sequence-to-sequence NMT model and investigate how using a syntactic tree helps the proposed model to resolve ambiguity in the input sentence as discussed in Section 3.1. Our proposed tree-to-sequence NMT model correctly translated the original source sentence "First, after static electricity of human body is released a Eathernet card is fixed." into "まず，人体の静電気を開放した後，イーサネットカードを装着した。("First, after static electricity of human body is released, an Eathernet card was fixed." in English). By using the English parser, the parsed tree clearly identifies that the source sentence is composed of two main large-scale phrase components of "after static electricity of human body is released" and "a Eathernet card is fixed." In order to investigate if feeding an obvious readable source sentence gives positive effect to the sequence-to-sequence NMT model, we manually fix the original source sentence in the three types of A, B, and C described in Table 10. However, the outputs are not correctly translated even after manually adding the comma after the word "released" to make the sentence boundary clear as shown in the source sentence A or after dynamically modifying the sentence structure as shown in the source sentences B and C.

6.4 Analysis of Attention Mechanism in Tree-to-Sequence NMT Model

Chinese-to-Japanese Translation. Figure 6 is a translation example on the development data set obtained by using our proposed model introduced in Section 5 and shows the attention relations between the source words or phrases and the target words with the

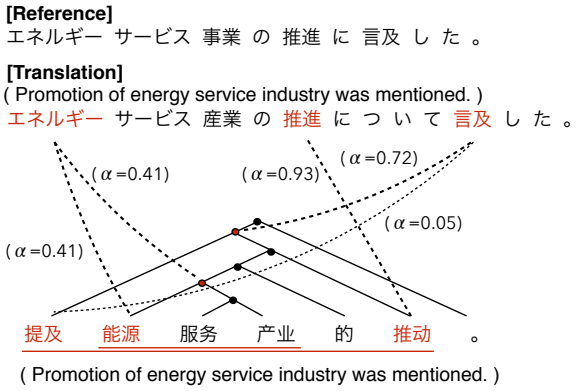

Figure 6
A translation example of a short sentence and the attention relations given by our proposed model.

highest attention score α. The source Chinese sentence is represented as a binary tree. We let our trained model generate a translated sentence in Japanese. Some target words are related to the source phrases with higher attention scores than the source words; for instance, the first generated target Japanese word "エネルギー (energy)" is related to the phrase "能源 服务 产业 (promotion of energy industry)" with a higher attention score $\alpha = 0.66$ than the translated word "能源 (energy)" with $\alpha = 0.22$. Our attention mechanism also aligns one Japanese word "推進 (promotion)" with the source word "推动" with the highest attention score $\alpha = 0.95$, and we can see that the attention mechanism in our proposed model flexibly learns which of the source words and phrases should be attended to more.

When generating a sentence, the attention mechanism provides the attention scores to the source units. We therefore counted the source targets with the five highest attention scores at each generated word to verify which of the source words or phrases are attended to more frequently in the Chinese-to-Japanese translation task. Table 12 shows the trends of the attention destinations toward either the source words or source phrases in both translation tasks. In the case of the Chinese-to-Japanese translation task,

Table 12
Percentages of the destinations of the attention mechanism to source words (Wrd) or phrases (Phr) by using the proposed tree-to-sequence NMT models described in Sections 5 and 6. The percentages are obtained on the Chinese-to-Japanese and English-to-Japanese development data, respectively.

	Chinese-to-Japanese		English-to-Japanese			
			Small Data set		Large Data set	
	Wrd (%)	Phr (%)	Wrd (%)	Phr (%)	Wrd (%)	Phr (%)
$\alpha > 0.2$	18.9	5.5	11.2	10.8	9.1	8.1
$\alpha > 0.4$	10.6	2.4	5.3	3.6	3.4	2.3
$\alpha > 0.6$	6.1	1.0	2.9	2.5	1.5	0.5
$\alpha > 0.8$	2.6	0.3	1.3	0.8	0.3	0.0

there exist a total of the 71,422 generated words on the development data. Here, the source words dominate the attention destinations rather than the source phrases at any thresholds of α with large difference. Our trained model uses the syntactic information but pays more attention to source words than source phrases. One possible reason is that both Chinese and Japanese share many Chinese characters, and thus it would be easy to perform word-by-word translation by aligning words of the same meanings.

We utilized the tree-to-sequence NMT models described in Section 5 with the BlackOut sampling method ($K = 2{,}000$) and in Section 6 with $K = 2{,}500$ for the English-to-Japanese translation. Here, the total numbers of generated words on the development data are 48,169 and 47,869 words by using the NMT models trained on the small training data and the large training data described in Table 2, respectively. Compared to the Chinese-to-Japanese translation task, the percentages of the source phrases as a destination become close to the source words at any $\alpha > n (n = 0.2, 0.4, 0.6, 0.8)$ within 2% for both cases. The attention mechanism in the English-to-Japanese translation is more frequently used to pay attention to the source phrases. As the training data increase, we can see that the source phrases are less likely to be paid attention to at each threshold of α.

7. Related Work

Kalchbrenner and Blunsom (2013) were the first to propose an end-to-end NMT model using convolutional neural networks (CNNs) for a source encoder and using RNNs for a target decoder. The encoder-decoder model can be seen as an extension of their model, and it replaces the CNNs with RNNs using GRU units (Cho et al. 2014b) or LSTM units (Sutskever, Vinyals, and Le 2014). Sutskever, Vinyals, and Le (2014) have shown that making the input sequences reversed is effective in a French-to-English translation task, and the technique has also proven effective in translation tasks between other European language pairs (Luong, Pham, and Manning 2015). All of the NMT models mentioned here are based on sequential encoders. The attention mechanism (Bahdanau, Cho, and Bengio 2015) has promoted the NMT models onto the next stage so that NMT models generate longer translated sentences by softly aligning the target words with the source words. Luong, Pham, and Manning (2015) refined the attention model so that it can dynamically focus on local windows rather than the entire sentence. They also proposed a more effective attention path in the calculation of the NMT models.

To incorporate structural information into the NMT models, Cho et al. (2014a) proposed to jointly learn structures inherent in source-side languages but did not report improvement of translation performance. These studies motivated us to investigate the role of syntactic structures explicitly given by existing syntactic parsers in the NMT models. Incorporating the syntactic structures into the NMT models has been actively studied recently. Chen et al. (2017) have extended our work by using bi-directional RNNs and RvNNs. Bastings et al. (2017) apply graph CNNs to encode the dependency trees. The previous work relies on syntactic trees provided by parsers and has a demerit of introducing parse errors into the models, whereas another trend is to learn and enhance latent structures or relations between any input units in source sentences in the NMT models (Bradbury and Socher 2017; Vaswani et al. 2017; Yoon Kim and Rush 2017). Hashimoto and Tsuruoka (2017) and Tran and Bisk (2018) reported an interesting observation that the latent parsed trees learned through the training are not always consistent with those obtained by the existing parsers, which suggests that there might exist a favorable structure of a source sentence depending on a task of interest.

Although it is relatively easy to encode a source syntactic tree into a vector space, decoding a target sentence with a parse tree is known as a challenging task and has been recently explored in sentence generation tasks (Dong and Lapata 2016) and language modeling tasks (Dyer et al. 2016). Wu et al. (2017) and Eriguchi, Tsuruoka, and Cho (2017) have proposed hybrid models that jointly learn to parse and translate by using target-side dependency trees, and Aharoni and Goldberg (2017) serialize a target parse tree to a sequence of units in order to apply it to the sequence-to-sequence NMT model. As we can see in the recent active studies of syntax-based approaches in the MT area, we believe that incorporating structural biases into NLP models is a promising research direction.

8. Conclusion

We propose a syntactic approach that extends the existing sequence-to-sequence NMT models. We focus on source-side phrase structures and build a tree-based encoder following the parse trees. Our proposed tree-based encoder is a natural extension of the sequential encoder model, where the leaf units of the tree-LSTM in the encoder can work together with the original sequential LSTM encoder. Moreover, the proposed attention mechanism allows the tree-based encoder to align not only the input words but also the input phrases with the output words. Experimental results show that the English-to-Japanese translation task benefits from incorporating syntactic trees more than the Chinese-to-Japanese translation task, and using the bi-directional encoder better improves the translation accuracy and achieves the best scores in both tasks. As the training data set becomes larger, we observed that the tree-to-sequence gives the smaller improvements on the WAT 2015 English-to-Japanese translation task. Our analyses on the tree-to-sequence models reveal different trends with the sequence-to-sequence models, and we have also shown that our proposed model flexibly learns which source words or phrases are attended to.

Acknowledgments

This work was supported by JST CREST grants JPMJCR1513 and JSPS KAKENHI grants 15J12597, 16H01715, and 17J09620.

References

Aharoni, Roee and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations 2015*, San Diego, CA.

Bastings, Joost, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen.

Bradbury, James and Richard Socher. 2017. Towards neural machine translation with latent tree attention. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 12–16, Copenhagen.

Chen, Huadong, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945, Copenhagen.

Cho, Kyunghyun, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, pages 103–111, Doha.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Dong, Li and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.

Dyer, Chris, Adhiguna Kuncoro, Miguel Ballesteros, and A. Noah Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209.

Elman, Jeffrey L. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Eriguchi, Akiko, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 823–833, Berlin.

Eriguchi, Akiko, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78, Vancouver.

Gehring, Jonas, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, Sydney.

Gers, Felix A., Jürgen Schmidhuber, and Fred A. Cummins. 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Gutmann, Michael U. and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(1):307–361.

Hashimoto, Kazuma and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 125–135, Copenhagen.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Isozaki, Hideki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952, Cambridge, MA.

Ji, Shihao, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2016. BlackOut: Speeding up recurrent neural network language models with very large vocabularies. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, PR.

Józefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 2342–2350, Lille.

Kalchbrenner, Nal and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, WA.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 388–395, Barcelona.

Koehn, Philipp and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Lee, Hyoung Gyu, JaeSong Lee, Jun-Seok Kim, and Chang-Ki Lee. 2015. NAVER machine translation system for WAT 2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 69–73, Kyoto.

Levy, Roger and Christopher D. Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 439–446, Sapporo.

Liu, Yang, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and*

44th Annual Meeting of the Association for Computational Linguistics, pages 609–616.

Luong, Thang, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Beijing.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, Lake Tahoe, CA.

Miyao, Yusuke and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Nakazawa, Toshiaki, Hideya Mino, Isao Goto, Graham Neubig, Sadao Kurohashi, and Eiichiro Sumita. 2015. Overview of the 2nd Workshop on Asian Translation. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 1–28, Kyoto.

Nakazawa, Toshiaki, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian Scientific Paper Excerpt Corpus. In *Proceedings of the 10th Conference on International Language Resources and Evaluation*, pages 2204–2208, Portorož.

Neubig, Graham and Kevin Duh. 2014. On the elements of an accurate tree-to-string machine translation system. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–149, Baltimore, MD.

Neubig, Graham, Makoto Morishita, and Satoshi Nakamura. 2015. Neural reranking improves subjective quality of machine translation: NAIST at WAT2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 35–41, Kyoto.

Neubig, Graham, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533, Portland, OR.

Oda, Yusuke, Philip Arthur, Graham Neubig, Koichiro Yoshino, and Satoshi Nakamura. 2017. Neural machine translation via binary code prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 850–860, Vancouver.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *arXiv: 1211.5063*.

Pollack, J. B. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105.

Pouget-Abadie, Jean, Dzmitry Bahdanau, Bart van Merrienboer, Kyunghyun Cho, and Yoshua Bengio. 2014. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 78–85, Doha.

Qiao, Yuchen, Kazuma Hashimoto, Akiko Eriguchi, Haixia Wang, Dongsheng Wang, Yoshimasa Tsuruoka, and Kenjiro Taura. 2017. Cache friendly parallelization of neural encoder-decoder models without padding on multi-core architecture. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2017*, pages 437–440, Lake Buena Vista, FL.

Sag, Ivan A., Thomas Wasow, and Emily Bender. 2003. *Syntactic Theory: A Formal Introduction*, 2nd edition. Center for the Study of Language and Information.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, 3104–3112, Montreal.

Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing.

Tran, Ke and Yonatan Bisk. 2018. Inducing grammars with and for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 25–35, Melbourne.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, pages 5998–6008, Long Beach, CA.

Wu, Shuangzhi, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707, Vancouver.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

Yoon, Kim, Carl Denton, Luong Hoang and Alexander M. Rush. 2017. Structured attention networks. In *Proceedings of International Conference on Learning Representations 2017*, Toulon.

Zhou, Jie, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383.

Zhu, Zhongyuan. 2015. Evaluating Neural Machine Translation in English-Japanese Task. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 61–68.