Tree-to-Sequence Attentional Neural Machine Translation

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka
The University of Tokyo, 3-7-1 Hongo, Bunkyo-ku, Tokyo, Japan
{eriguchi, hassy, tsuruoka}@logos.t.u-tokyo.ac.jp

Abstract

Most of the existing Neural Machine Translation (NMT) models focus on the conversion of sequential data and do not directly use syntactic information. We propose a novel end-to-end syntactic NMT model, extending a sequence-to-sequence model with the source-side phrase structure. Our 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. Experimental results on the WAT'15 English-to-Japanese dataset demonstrate that our proposed model considerably outperforms sequence-to-sequence attentional NMT models and compares favorably with the state-of-the-art tree-to-string SMT system.

1 Introduction

Machine Translation (MT) has traditionally been one of the most complex language processing problems, 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 et al., 2014), a Recurrent Neural Network (RNN) called the *encoder* reads the whole sequence of source words to produce a fixed-length vector, and then another RNN called the *decoder* generates the target words from the vector. The Encoder-Decoder model has been extended with an *attention* mechanism (Bahdanau et al., 2015; Luong et al., 2015a), which allows the model to jointly learn the soft alignment between the source language and the target language. NMT models have achieved state-of-the-art results in English-to-French and English-to-German translation tasks (Luong et al., 2015b; Luong et al., 2015a). However, it is yet to be seen whether NMT is competitive with traditional Statistical Machine Translation (SMT) approaches in translation tasks for structurally distant language pairs such as English-to-Japanese.

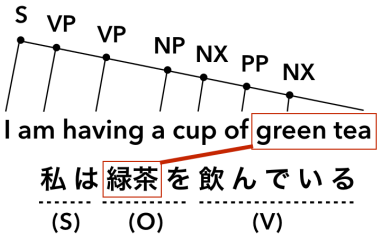

Figure 1: Alignment between an English phrase and a Japanese word.

Figure 1 shows a pair of parallel sentences in English and Japanese. English and Japanese are linguistically distant in many respects; they have different syntactic constructions, and words and phrases are defined in different lexical units. In this example, the Japanese word "緑茶" is aligned with the English words "green" and "tea", and the English word sequence "a cup of" is aligned with a special symbol "*null*", which 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 cup of green tea" with "緑茶". In SMT, it is known that incorporating syntactic constituents of the source language into the models improves word alignment (Yamada and Knight, 2001) and translation accuracy (Liu et al., 2006; Neubig and Duh, 2014). However, the existing NMT models do not allow us to perform this kind of alignment.

In this paper, we propose a novel attentional NMT model to take advantage of syntactic infor-

mation. Following the phrase structure of a source sentence, we encode the sentence recursively in a bottom-up fashion to produce a vector representation of the sentence and decode it while aligning the input phrases and words with the output. Our experimental results on the WAT'15 English-to-Japanese translation task show that our proposed model achieves state-of-the-art translation accuracy.

2 Neural Machine Translation

2.1 Encoder-Decoder Model

NMT is an end-to-end approach to data-driven machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). In other words, the NMT models directly estimate the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ given a large collection of source and target sentence pairs $(\boldsymbol{x}, \boldsymbol{y})$. An NMT model consists of an encoder process and a decoder process, and hence they are often called *Encoder-Decoder* models. In the Encoder-Decoder models, a sentence is treated as a sequence of words. In the encoder process, the encoder embeds each of the source words $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ into a d-dimensional vector space. The decoder then outputs a word sequence $\boldsymbol{y} = (y_1, y_2, \cdots, y_m)$ in the target language given the information on the source sentence provided by the encoder. Here, n and m are the lengths of the source and target sentences, respectively. RNNs allow one to effectively embed sequential data into the vector space.

In the RNN encoder, the i-th hidden unit $\boldsymbol{h}_i \in \mathbb{R}^{d\times 1}$ is calculated given the i-th input x_i and the previous hidden unit $\boldsymbol{h}_{i-1} \in \mathbb{R}^{d\times 1}$,

$$\boldsymbol{h}_i \;=\; f_{enc}(x_i, \boldsymbol{h}_{i-1}), \tag{1}$$

where f_{enc} is a non-linear function, and the initial hidden unit $\boldsymbol{h}_0$ is usually set to zeros. The encoding function f_{enc} is recursively applied until the n-th hidden unit $\boldsymbol{h}_n$ is obtained. The RNN Encoder-Decoder models assume that $\boldsymbol{h}_n$ represents a vector of the meaning of the input sequence up to the n-th word.

After encoding the whole input sentence into the vector space, we decode it in a similar way. The initial decoder unit $\boldsymbol{s}_1$ is initialized with the input sentence vector ($\boldsymbol{s}_1 = \boldsymbol{h}_n$). Given the previous target word and the j-th hidden unit of the decoder, the conditional probability that the j-th target word is generated is calculated as follows:

$$p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) \;=\; g(\boldsymbol{s}_j), \tag{2}$$

where g is a non-linear function. The j-th hidden unit of the decoder is calculated by using another non-linear function f_{dec} as follows:

$$\boldsymbol{s}_j \;=\; f_{dec}(y_{j-1}, \boldsymbol{s}_{j-1}). \tag{3}$$

We employ Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) in place of vanilla RNN units. The t-th LSTM unit consists of several *gates* and two different types of states: a hidden unit $\boldsymbol{h}_t \in \mathbb{R}^{d\times 1}$ and a memory cell $\boldsymbol{c}_t \in \mathbb{R}^{d\times 1}$,

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{U}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}),\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{U}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}),\\
\tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{W}^{(\tilde{c})}\boldsymbol{x}_t + \boldsymbol{U}^{(\tilde{c})}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(\tilde{c})}),\\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1},\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t),
\end{aligned} \tag{4}$$

where each of $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$ and $\tilde{\boldsymbol{c}}_t \in \mathbb{R}^{d\times 1}$ denotes an input gate, a forget gate, an output gate, and a state for updating the memory cell, respectively. $\boldsymbol{W}^{(\cdot)} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{U}^{(\cdot)} \in \mathbb{R}^{d\times d}$ are weight matrices, $\boldsymbol{b}^{(\cdot)} \in \mathbb{R}^{d\times 1}$ is a bias vector, and $\boldsymbol{x}_t \in \mathbb{R}^{d\times 1}$ is the word embedding of the t-th input word. $\sigma(\cdot)$ is the logistic function, and the operator $\odot$ denotes element-wise multiplication between vectors.

2.2 Attentional Encoder-Decoder Model

The NMT models with an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015a) have been proposed to softly align each decoder state with the encoder states. The attention mechanism allows the NMT models to explicitly quantify how much each encoder state contributes to the word prediction at each time step.

In the attentional NMT model in Luong et al. (2015a), at the j-th step of the decoder process, the attention score $\boldsymbol{\alpha}_j(i)$ between the i-th source hidden unit $\boldsymbol{h}_i$ and the j-th target hidden unit $\boldsymbol{s}_j$ is calculated as follows:

$$\boldsymbol{\alpha}_j(i) \;=\; \frac{\exp(\boldsymbol{h}_i \cdot \boldsymbol{s}_j)}{\sum_{k=1}^{n}\exp(\boldsymbol{h}_k \cdot \boldsymbol{s}_j)}, \tag{5}$$

where $\boldsymbol{h}_i \cdot \boldsymbol{s}_j$ is the inner product of $\boldsymbol{h}_i$ and $\boldsymbol{s}_j$, which is used to directly calculate the similarity score between $\boldsymbol{h}_i$ and $\boldsymbol{s}_j$. The j-th context vector

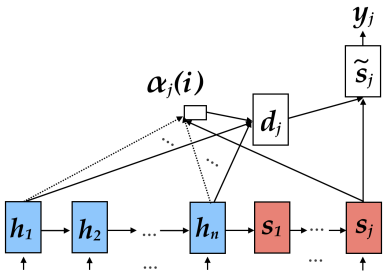

Figure 2: Attentional Encoder-Decoder model.

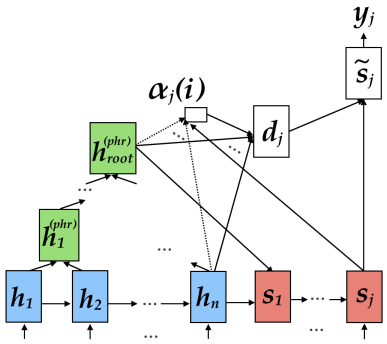

Figure 3: Proposed model: Tree-to-sequence attentional NMT model.

$\boldsymbol{d}_j$ is calculated as the summation vector weighted by $\boldsymbol{\alpha}_j(i)$:

$$\boldsymbol{d}_j = \sum_{i=1}^{n} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i. \quad (6)$$

To incorporate the attention mechanism into the decoding process, the context vector is used for the the j-th word prediction by putting an additional hidden layer $\tilde{\boldsymbol{s}}_j$:

$$\tilde{\boldsymbol{s}}_j = \tanh(\boldsymbol{W}_d[\boldsymbol{s}_j;\ \boldsymbol{d}_j] + \boldsymbol{b}_d), \quad (7)$$

where $[\boldsymbol{s}_j; \boldsymbol{d}_j] \in \mathbb{R}^{2d\times 1}$ is the concatenation of $\boldsymbol{s}_j$ and $\boldsymbol{d}_j$, and $\boldsymbol{W}_d \in \mathbb{R}^{d\times 2d}$ and $\boldsymbol{b}_d \in \mathbb{R}^{d\times 1}$ are a weight matrix and a bias vector, respectively. The model predicts the j-th word by using the softmax function:

$$p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_s\tilde{\boldsymbol{s}}_j + \boldsymbol{b}_s), \quad (8)$$

where $\boldsymbol{W}_s \in \mathbb{R}^{|V|\times d}$ and $\boldsymbol{b}_s \in \mathbb{R}^{|V|\times 1}$ are a weight matrix and a bias vector, respectively. $|V|$ stands for the size of the vocabulary of the target language. Figure 2 shows an example of the NMT model with the attention mechanism.

2.3 Objective Function of NMT Models

The objective function to train the NMT models is the sum of the log-likelihoods of the translation pairs in the training data:

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

where $\mathcal{D}$ denotes a set of parallel sentence pairs. The model parameters $\boldsymbol{\theta}$ are learned through Stochastic Gradient Descent (SGD).

3 Attentional Tree-to-Sequence Model

3.1 Tree-based Encoder + Sequential Encoder

The exsiting NMT models treat a sentence as a sequence of words and neglect the structure of a sentence inherent in language. We propose a novel tree-based encoder in order to explicitly take the syntactic structure into consideration in the NMT model. We focus on the phrase structure of a sentence and construct a sentence vector from phrase vectors in a bottom-up fashion. The sentence vector in the tree-based encoder is therefore composed of the structural information rather than the sequential data. Figure 3 shows our proposed model, which we call a *tree-to-sequence attentional NMT model*.

In Head-driven Phrase Structure Grammar (HPSG) (Sag et al., 2003), a sentence is composed of multiple phrase units and represented as a binary tree as shown in Figure 1. Following the structure of the sentence, we construct a tree-based encoder on top of the standard sequential encoder. The k-th parent hidden unit $\boldsymbol{h}_k^{(phr)}$ for the k-th phrase is calculated using the left and right child hidden units $\boldsymbol{h}_k^l$ and $\boldsymbol{h}_k^r$ as follows:

$$\boldsymbol{h}_k^{(phr)} = f_{tree}(\boldsymbol{h}_k^l, \boldsymbol{h}_k^r), \quad (10)$$

where f_{tree} is a non-linear function.

We construct a tree-based encoder with LSTM units, where each node in the binary tree is represented with an LSTM unit. When initializing the leaf units of the tree-based encoder, we employ the sequential LSTM units described in Section 2.1. Each non-leaf node is also represented with an LSTM unit, and we employ Tree-LSTM (Tai et al., 2015) to calculate the LSTM unit of the parent node which has two child LSTM units. The hidden unit $\boldsymbol{h}_k^{(phr)} \in \mathbb{R}^{d\times 1}$ and the memory cell $\boldsymbol{c}_k^{(phr)} \in \mathbb{R}^{d\times 1}$ for the k-th parent node are calcu-

lated as follows:

$$\begin{aligned}
\boldsymbol{i}_k &= \sigma(\boldsymbol{U}_l^{(i)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(i)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_k^l &= \sigma(\boldsymbol{U}_l^{(f_l)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_l)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(f_l)}),\\
\boldsymbol{f}_k^r &= \sigma(\boldsymbol{U}_l^{(f_r)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_r)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(f_r)}),\\
\boldsymbol{o}_k &= \sigma(\boldsymbol{U}_l^{(o)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(o)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(o)}),\\
\tilde{\boldsymbol{c}}_k &= \tanh(\boldsymbol{U}_l^{(\tilde{c})}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(\tilde{c})}\boldsymbol{h}_k^r + \boldsymbol{b}^{(\tilde{c})}),\\
\boldsymbol{c}_k^{(phr)} &= \boldsymbol{i}_k \odot \tilde{\boldsymbol{c}}_k + \boldsymbol{f}_k^l \odot \boldsymbol{c}_k^l + \boldsymbol{f}_k^r \odot \boldsymbol{c}_k^r,\\
\boldsymbol{h}_k^{(phr)} &= \boldsymbol{o}_k \odot \tanh(\boldsymbol{c}_k^{(phr)}), \qquad (11)
\end{aligned}$$

where $\boldsymbol{i}_k$, $\boldsymbol{f}_k^l$, $\boldsymbol{f}_k^r$, $\boldsymbol{o}_j$, $\tilde{\boldsymbol{c}}_j \in \mathbb{R}^{d\times 1}$ are an input gate, the forget gates for left and right child units, an output gate, and a state for updating the memory cell, respectively. $\boldsymbol{c}_k^l$ and $\boldsymbol{c}_k^r$ are the memory cells for the left and right child units, respectively. $\boldsymbol{U}^{(\cdot)} \in \mathbb{R}^{d\times d}$ denotes a weight matrix, and $\boldsymbol{b}^{(\cdot)} \in \mathbb{R}^{d\times 1}$ represents a bias vector.

Our proposed tree-based encoder is a natural extension of the conventional sequential encoder, since Tree-LSTM is a generalization of chain-structured LSTM (Tai et al., 2015). Our encoder differs from the original Tree-LSTM in the calculation of the LSTM units for the leaf 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 LSTMs are calculated in the context of the previous units. This ability contrasts with the original Tree-LSTM, in which the leaves are composed only of the word embeddings without any contextual information.

3.2 Initial Decoder Setting

We now have two different sentence vectors: one is from the sequence encoder and the other from the tree-based encoder. As shown in Figure 3, we provide another Tree-LSTM unit which has the final sequential encoder unit ($\boldsymbol{h}_n$) and the tree-based encoder unit ($\boldsymbol{h}_{root}^{(phr)}$) as two child units and set it as the initial decoder $\boldsymbol{s}_1$ as follows:

$$\boldsymbol{s}_1 = g_{tree}(\boldsymbol{h}_n, \boldsymbol{h}_{root}^{(phr)}), \qquad (12)$$

where g_{tree} is the same function as f_{tree} with another set of Tree-LSTM parameters. This initialization allows the decoder to capture information from both the sequential data and phrase structures. Zoph and Knight (2016) proposed a similar method using a Tree-LSTM for initializing the decoder, with which they translate multiple source languages to one target language. When the syntactic parser fails to output a parse tree for a sentence, we encode the sentence with the sequential encoder by setting $\boldsymbol{h}_{root}^{(phr)} = \boldsymbol{0}$. Our proposed tree-based encoder therefore works with any sentences.

3.3 Attention Mechanism in Our Model

We adopt the attention mechanism into our tree-to-sequence model in a novel way. Our model gives attention not only to sequential hidden units but also to phrase hidden units. This attention mechanism tells us which words or phrases in the source sentence are important when the model decodes a target word. The j-th context vector $\boldsymbol{d}_j$ is composed of the sequential and phrase vectors weighted by the attention score $\boldsymbol{\alpha}_j(i)$:

$$\boldsymbol{d}_j = \sum_{i=1}^{n} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i + \sum_{i=n+1}^{2n-1} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i^{(phr)}. \qquad (13)$$

Note that a binary tree has $n-1$ phrase nodes if the tree has n leaves. We set a final decoder $\tilde{\boldsymbol{s}}_j$ in the same way as Equation (7).

In addition, we adopt the *input-feeding* method (Luong et al., 2015a) in our model, which is a method for feeding $\tilde{\boldsymbol{s}}_{j-1}$, the previous unit to predict the word y_{j-1}, into the current target hidden unit $\boldsymbol{s}_j$,

$$\boldsymbol{s}_j = f_{dec}(y_{j-1}, [\boldsymbol{s}_{j-1};\ \tilde{\boldsymbol{s}}_{j-1}]), \qquad (14)$$

where $[\boldsymbol{s}_{j-1};\ \tilde{\boldsymbol{s}}_{j-1}]$ is the concatenation of $\boldsymbol{s}_{j-1}$ and $\tilde{\boldsymbol{s}}_{j-1}$. The input-feeding approach contributes to the enrichment in the calculation of the decoder, because $\tilde{\boldsymbol{s}}_{j-1}$ is an informative unit which can be used to predict the output word as well as to be compacted with attentional context vectors. Luong et al. (2015a) showed that the input-feeding approach improves BLEU scores. We also observed the same improvement in our preliminary experiments.

3.4 Sampling-Based Approximation to the NMT Models

The biggest computational bottleneck of training the NMT models is in the calculation of the softmax layer described in Equation (8), because its computational cost increases linearly with the size of the vocabulary. The speedup technique with GPUs has proven useful for sequence-based NMT models (Sutskever et al., 2014; Luong et al.,

2015a) but it is not easily applicable when dealing with tree-structured data. In order to reduce the training cost of the NMT models at the softmax layer, we employ *BlackOut* (Ji et al., 2016), a sampling-based approximation method. BlackOut has been shown to be effective in RNN Language Models (RNNLMs) and allows a model to run reasonably fast even with a million word vocabulary with CPUs.

At each word prediction step in the training, BlackOut estimates the conditional probability in Equation (2) for the target word and K negative samples using a weighted softmax function. The negative samples are drawn from the unigram distribution raised to the power $\beta \in [0, 1]$ (Mikolov et al., 2013). The unigram distribution is estimated using the training data and β is a hyperparameter. BlackOut is closely related to Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2012) and achieves better perplexity than the original softmax and NCE in RNNLMs. The advantages of Blackout over the other methods are discussed in Ji et al. (2016). Note that BlackOut can be used as the original softmax once the training is finished.

4 Experiments

4.1 Training Data

We applied the proposed model to the English-to-Japanese translation dataset of the ASPEC corpus given in WAT'15.[1] Following Zhu (2015), we extracted the first 1.5 million translation pairs from the training data. To obtain the phrase structures of the source sentences, i.e., English, we used the probabilistic HPSG parser *Enju* (Miyao and Tsujii, 2008). We used Enju only to obtain a binary phrase structure for each sentence and did not use any HPSG specific information. For the target language, i.e., Japanese, we used KyTea (Neubig et al., 2011), a Japanese segmentation tool, and performed the pre-processing steps recommended in WAT'15.[2] We then filtered out the translation pairs whose sentence lengths are longer than 50 and whose source sentences are not parsed successfully. Table 1 shows the details of the datasets used in our experiments. We carried out two experiments on a small training dataset to investigate the effectiveness of our proposed model and on a large training dataset to compare our proposed methods with the other systems.

	Sentences	Parsed successfully
Train	1,346,946	1,346,946
Development	1,790	1,789
Test	1,812	1,811

Table 1: Dataset in ASPEC corpus.

	Train (small)	Train (large)
sentence pairs	100,000	1,346,946
$\|V\|$ in English	25,478	87,796
$\|V\|$ in Japanese	23,532	65,680

Table 2: Training dataset and the vocabulary sizes.

The vocabulary consists of words observed in the training data more than or equal to N times. We set $N = 2$ for the small training dataset and $N = 5$ for the large training dataset. The out-of-vocabulary words are mapped to the special token "*unk*". We added another special symbol "*eos*" for both languages and inserted it at the end of all the sentences. Table 2 shows the details of each training dataset and its corresponding vocabulary size.

4.2 Training Details

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). Following Józefowicz et al. (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. We halve the learning rate when the development loss becomes worse. Gradient norms are clipped to 3.0 to avoid exploding gradient problems (Pascanu et al., 2012).

Small Training Dataset We conduct experiments with our proposed model and the sequential attentional NMT model with the input-feeding approach. Each model has 256-dimensional hidden units and word embeddings. The number of negative samples K of BlackOut is set to 500 or 2000.

[1] http://orchid.kuee.kyoto-u.ac.jp/WAT/WAT2015/index.html

[2] http://orchid.kuee.kyoto-u.ac.jp/WAT/WAT2015/baseline/dataPreparationJE.html

Large Training Dataset Our proposed model has 512-dimensional word embeddings and d-dimensional hidden units ($d \in \{512, 768, 1024\}$). K is set to 2500.

Our code[3] is implemented in C++ using the Eigen library,[4] a template library for linear algebra, and we run all of the experiments on multi-core CPUs.[5] It takes about a week to train a model on the large training dataset with $d = 512$.

4.3 Decoding process

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

$$score(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{m} \log p(y_j | \boldsymbol{y}_{<j}, \boldsymbol{x}). \quad (15)$$

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 (Cho et al., 2014a; Pouget-Abadie et al., 2014). In our preliminary experiments, the beam search with the length normalization in Cho et al. (2014a) was not effective in English-to-Japanese translation. The method in Pouget-Abadie et al. (2014) needs to estimate the conditional probability $p(\boldsymbol{x}|\boldsymbol{y})$ using another NMT model and thus is not suitable for our work.

In this paper, we use statistics on sentence lengths in beam search. Assuming that the length of a target sentence correlates with the length of a source sentence, we redefine the score of each candidate as follows:

$$score(\boldsymbol{x}, \boldsymbol{y}) = L_{\boldsymbol{x},\boldsymbol{y}} + \sum_{j=1}^{m} \log p(y_j | \boldsymbol{y}_{<j}, \boldsymbol{x}), \quad (16)$$

$$L_{\boldsymbol{x},\boldsymbol{y}} = \log p(len(\boldsymbol{y}) | len(\boldsymbol{x})), \quad (17)$$

where $L_{\boldsymbol{x},\boldsymbol{y}}$ is the penalty for the conditional probability of the target sentence length $len(\boldsymbol{y})$ given the source sentence length $len(\boldsymbol{x})$. It allows the model to decode a sentence by considering the length of the target sentence. In our experiments, we computed the conditional probability $p(len(\boldsymbol{y})|len(\boldsymbol{x}))$ in advance following the statistics collected in the first one million pairs of the training dataset. We allow the decoder to generate up to 100 words.

4.4 Evaluation

We evaluated the models by two automatic evaluation metrics, RIBES (Isozaki et al., 2010) and BLEU (Papineni et al., 2002) following WAT'15. We used the KyTea-based evaluation script for the translation results.[6] The RIBES score is a metric based on rank correlation coefficients with word precision, and the BLEU score is based on n-gram word precision and a Brevity Penalty (BP) for outputs shorter than the references. RIBES is known to have stronger correlation with human judgements than BLEU in translation between English and Japanese as discussed in Isozaki et al. (2010).

5 Results and Discussion

5.1 Small Training Dataset

Table 3 shows the perplexity, BLEU, RIBES, and the training time on the development data with the Attentional NMT (ANMT) models trained on the small dataset. We conducted the experiments with our proposed method using BlackOut and softmax. We decoded a translation by our proposed beam search with a beam size of 20.

As shown in Table 3, the results of our proposed model with BlackOut improve as the number of negative samples K increases. Although the result of softmax is better than those of BlackOut ($K = 500, 2000$), the training time of softmax per epoch is about three times longer than that of BlackOut even with the small dataset.

As to the results of the ANMT model, reversing the word order in the input sentence decreases the scores in English-to-Japanese translation, which contrasts with the results of other language pairs reported in previous work (Sutskever et al., 2014; Luong et al., 2015a). By taking syntactic information into consideration, our proposed model improves the scores, compared to the sequential attention-based approach.

We 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

[3] https://github.com/tempra28/tree2seq

[4] http://eigen.tuxfamily.org/index.php

[5] 16 threads on Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz

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

	K	Perplexity	RIBES	BLEU	Time/epoch (min.)
Proposed model	500	19.6	71.8	20.0	55
Proposed model	2000	21.0	72.6	20.5	70
Proposed model (Softmax)	—	17.9	73.2	21.8	180
ANMT (Luong et al., 2015a)	500	21.6	70.7	18.5	45
+ reverse input	500	22.6	69.8	17.7	45
ANMT (Luong et al., 2015a)	2000	23.1	71.5	19.4	60
+ reverse input	2000	26.1	69.5	17.5	60

Table 3: Evaluation results on the development data using the small training data. The training time per epoch is also shown, and K is the number of negative samples in BlackOut.

	Beam size	RIBES	BLEU (BP)
Simple BS	6	72.3	20.0 (90.1)
	20	72.3	19.5 (85.1)
Proposed BS	20	**72.6**	**20.5** (91.7)

Table 4: Effects of the Beam Search (BS) on the development data.

	RIBES	BLEU
Without sequential LSTMs	69.4	19.5
With sequential LSTMs	**72.3**	**20.0**

Table 5: Effects of the sequential LSTMs in our proposed tree-based encoder on the development data.

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

Effects of the proposed beam search Table 4 shows the results on the development data of proposed method with BlackOut ($K = 2000$) by the simple beam search and our proposed beam search. The beam size is set to 6 or 20 in the simple beam search, and to 20 in our proposed search. We can see that our proposed search outperforms the simple beam search in both scores. Unlike RIBES, the BLEU score is sensitive to the beam size and becomes lower as the beam size increases. We found that the BP had a relatively large 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.

Effects of the sequential LSTM units We also investigated the effects of the sequential LSTMs at the leaf nodes in our proposed tree-based encoder. Table 5 shows the result on the development data of our proposed encoder and that of an attentional tree-based encoder without sequential LSTMs with BlackOut ($K = 2000$).[7] The results show that our proposed encoder considerably outperforms the encoder without sequential LSTMs, suggesting that the sequential LSTMs at the leaf nodes contribute to the context-aware construction of the phrase representations in the tree.

5.2 Large Training Dataset

Table 6 shows the experimental results of RIBES and BLEU scores achieved by the trained models on the large dataset. We decoded the target sentences by our proposed beam search with the beam size of 20.[8] 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 1024, and it shows the best RIBES score among all systems.[9]

As for the time required for training, our implementation needs about one day to perform one epoch on the large training dataset with $d = 512$. It would take about 11 days without using the BlackOut sampling.

Comparison with the NMT models The model of Zhu (2015) is an ANMT model (Bahdanau et al., 2015) with a bi-directional LSTM encoder, and uses 1024-dimensional hidden units and 1000-

[7] For this evaluation, we used the 1,789 sentences that were successfully parsed by Enju because the encoder without sequential LSTMs always requires a parse tree.

[8] We found two sentences which ends without *eos* with $d = 512$, and then we decoded it again with the beam size of 1000 following Zhu (2015).

[9] Our ensemble model yields a METEOR (Denkowski and Lavie, 2014) score of 53.6 with language option "-l other".

Model	RIBES	BLEU
Proposed model ($d = 512$)	81.46	34.36
Proposed model ($d = 768$)	81.89	34.78
Proposed model ($d = 1024$)	81.58	34.87
Ensemble of the above three models	**82.45**	36.95
ANMT with LSTMs (Zhu, 2015)	79.70	32.19
+ Ensemble, *unk* replacement	80.27	34.19
+ System combination, 3 pre-reordered ensembles	80.91	36.21
ANMT with GRUs (Lee et al., 2015) + character-based decoding, Begin/Inside representation	81.15	35.75
PB baseline	69.19	29.80
HPB baseline	74.70	32.56
T2S baseline	75.80	33.44
T2S model (Neubig and Duh, 2014)	79.65	36.58
+ ANMT Rerank (Neubig et al., 2015)	81.38	38.17

Table 6: Evaluation results on the test data.

dimensional word embeddings. The model of Lee et al. (2015) is also an ANMT model with a bi-directional Gated Recurrent Unit (GRU) encoder, and uses 1000-dimensional hidden units and 200-dimensional word embeddings. Both models are sequential ANMT models. Our single proposed model with $d = 512$ outperforms the best result of Zhu (2015)'s end-to-end NMT model with ensemble and unknown replacement by $+1.19$ RIBES and by $+0.17$ BLEU scores. Our ensemble model shows better performance, in both RIBES and BLEU scores, than that of Zhu (2015)'s best system which is a hybrid of the ANMT and SMT models by $+1.54$ RIBES and by $+0.74$ BLEU scores and Lee et al. (2015)'s ANMT system with special character-based decoding by $+1.30$ RIBES and $+1.20$ BLEU scores.

Comparison with the SMT models PB, HPB and T2S are the baseline SMT systems in WAT'15: a phrase-based model, a hierarchical phrase-based model, and a tree-to-string model, respectively (Nakazawa et al., 2015). The best model in WAT'15 is Neubig et al. (2015)'s tree-to-string SMT model enhanced with reranking by ANMT using a bi-directional LSTM encoder. Our proposed end-to-end NMT model compares favorably with Neubig et al. (2015).

5.3 Qualitative Analysis

We illustrate the translations of test data by our model with $d = 512$ and several attentional relations when decoding a sentence. In Figures 4 and 5, an English sentence represented as a binary tree is translated into Japanese, and several attentional relations between English words or phrases and

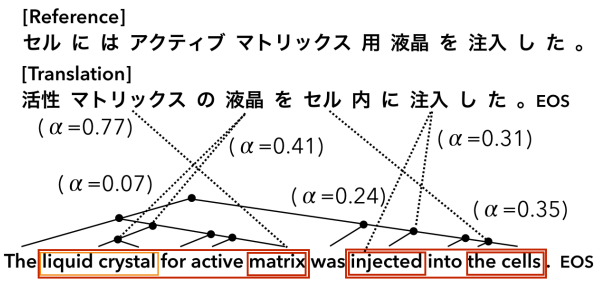

Figure 4: Translation example of a short sentence and the attentional relations by our proposed model.

Japanese word are shown with the highest attention score α. The additional attentional relations are also illustrated for comparison. We can see 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* described as 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 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 attentional 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

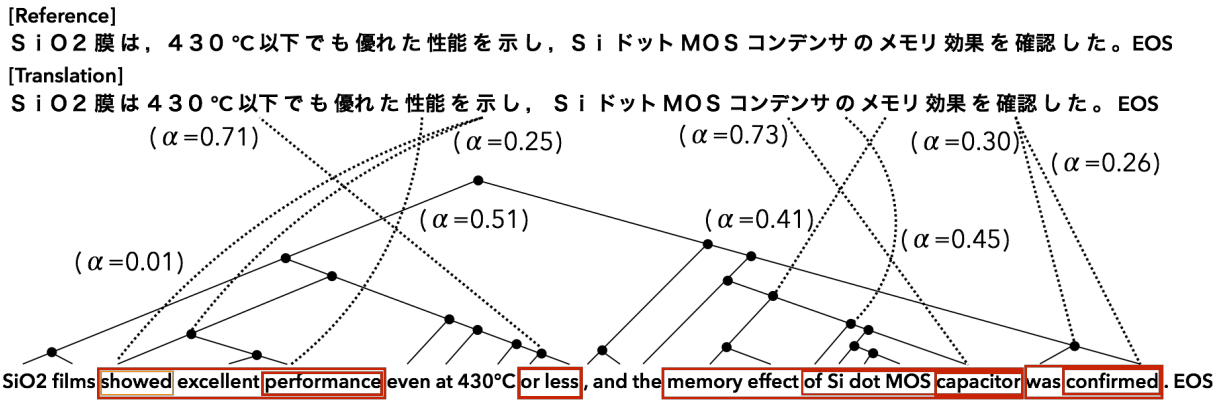

Figure 5: Translation example of a long sentence and the attentional relations by our proposed model.

synonyms of the reference words, e.g. “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, 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’15 by crowd sourcing (Nakazawa et al., 2015).

6 Related Work

Kalchbrenner and Blunsom (2013) were the first to propose an end-to-end NMT model using Convolutional Neural Networks (CNNs) as the source encoder and using RNNs as the target decoder. The Encoder-Decoder model can be seen as an extension of their model, and it replaces the CNNs with RNNs using GRUs (Cho et al., 2014b) or LSTMs (Sutskever et al., 2014).

Sutskever et al. (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 et al., 2015a). All of the NMT models mentioned above are based on sequential encoders. 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.

The attention mechanism (Bahdanau et al., 2015) has promoted NMT onto the next stage. It enables the NMT models to translate while aligning the target with the source. Luong et al. (2015a) refined the attention model so that it can dynamically focus on local windows rather than the entire sentence. They also proposed a more effective attentional path in the calculation of ANMT models. Subsequently, several ANMT models have been proposed (Cheng et al., 2016; Cohn et al., 2016); however, each model is based on the existing sequential attentional models and does not focus on a syntactic structure of languages.

7 Conclusion

In this paper, we propose a novel syntactic approach that extends attentional NMT models. We focus on the phrase structure of the input sentence and build a tree-based encoder following the parsed tree. 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 attention mechanism allows the tree-based encoder to align not only the input words but also input phrases with the output words. Experimental results on the WAT’15 English-to-Japanese translation dataset demonstrate that our proposed model achieves the best RIBES score and outperforms the sequential attentional NMT model.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions. This work was supported by CREST, JST, and JSPS KAKENHI Grant Number 15J12597.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Yong Cheng, Shiqi Shen, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Agreement-based Joint Training for Bidirectional Attention-based Neural Machine Translation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. to appear.

KyungHyun Cho, 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)*.

Kyunghyun Cho, 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.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating Structural Alignment Biases into an Attentional Neural Translation Model. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. to appear.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics 2014 Workshop on Statistical Machine Translation*.

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

Michael U. Gutmann 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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hideki Isozaki, 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.

Shihao Ji, 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*.

Rafal Józefowicz, 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.

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

Hyoung-Gyu Lee, 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.

Yang Liu, 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.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the Rare Word Problem in Neural Machine Translation. 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 11–19.

Tomas Mikolov, 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.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature Forest Models for Probabilistic HPSG Parsing. *Computational Linguistics*, 34(1):35–80.

Toshiaki Nakazawa, 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.

Graham Neubig 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.

Graham Neubig, 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.

Graham Neubig, 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.

Kishore Papineni, 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 on Association for Computational Linguistics*, pages 311–318.

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

Jean Pouget-Abadie, 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.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Kai Sheng Tai, 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.

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

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

Barret Zoph and Kevin Knight. 2016. Multi-Source Neural Translation. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. to appear.