博　　士　　論　　文

Incorporating Syntactic Structure into Neural Machine Translation

(構文構造に基づくニューラル機械翻訳)

指導教員　　鶴岡 慶雅 准教授

東京大学工学系研究科
電気系工学専攻

氏 名　　37-157068 江里口 瑛子

提 出 日　　平成29年12月1日

Abstract

Machine translation is one of the most difficult and complicated tasks in Natural Language Processing (NLP) because it is required to let a computer represent semantics of each language and convert the semantics from one language to another language so that humans can interpret the translations. Representation learning based on neural networks treats every language data such as word, phrase, and sentence as vectors. The vector representations allow us to easily compute the objective information and to apply the learned representations to the other NLP tasks.

Neural machine translation (NMT) has been developed in the trends of representation learning. The NMT model is an end-to-end neural network-based model to directly learn translations statistically from a large data set, and the NMT models have already achieved the state-of-the-art performance in several European languages and Asian languages. In comparison to the conventional statistical machine translation models, the NMT has the simpler architecture and powerful performance when there is a lot of training data.

Most of the existing NMT models are modeled as a sequence-to-sequence learning which learns the relations between the input sequences and the output sequences. They do not utilize any syntactic information inherited the languages. It is, however ,well known that syntactic structure helps to improve the machine translation models in a statistical machine translation areas when treating the translation between Japanese and English which are syntactically different languages. In this thesis, we focus on the syntactic structures inherited in languages and extended the existing NMT model to incorporate the syntactic structures. We have studied phrase structures in a source language and dependency tree structures in a target language.

Firstly, we employ the phrase structure in a source language and build a tree-based encoder to explicitly construct a phrase vector following the phrase structure. We call our proposed model "Tree-to-Sequence Neural Machine Translation model". Our proposed model also has an attention mechanism which softly aligns a target output with source word-based units as well as with source phrase-based units. Experimenting on an English-to-Japanese translation task, we evaluated the models by automatic evaluation metrics. We have confirmed that our proposed tree-to-sequence neural machine translation achieved better accuracy than the existing sequence-to-sequence NMT models and achieved the state-of-the-art performance in the RIBES score.

Secondly, we applied the character-based decoding method to the above described tree-to-sequence neural machine translation model. Although most of the NMT models have been developed as a word-based model, the models have the vocabulary coverage problem because of

low-frequent words and unknown words in the corpus. We also explored the effectiveness to utilize the phrase label category in the tree-based encoder. We conducted the experiments on the English-to-Japanese translation tasks, and we report the different trends between the word-based decoder model and the character-based decoder model.

Lastly, we have proposed a target-side syntax-based model called "Sequence-to-Tree Neural Machine Translation". We focus on the dependency relations between words in a target sentence as syntactic structure. Our proposed model generates a translation while parsing the translated sentence simultaneously. Experimenting on translation tasks for four language pairs, we have confirmed that our proposed model improved the model performance better than the existing NMT model in every four language pairs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Natural language processing (NLP) is a research area to study the languages we use which are called natural language in contraposition to computer language. As the technology of the computers and the development of the communication networks have progressed rapidly, the volume of language data has increased and the NLP tasks became more complicated. The recent progresses in NLP have been brought by the research results of machine learning area, a statistical approach to train a mathematical model by using a large data. Neural networks is one of the machine learning methods and has achieved great successes in variety of research areas of text data [Mikolov et al., 2013a; Chen and Manning, 2014; Wu et al., 2016; Andor et al., 2016], image data [Mikolov et al., 2013a; Krizhevsky et al., 2012; Goodfellow et al., 2014; Shrivastava et al., 2017], and speech data [Hannun et al., 2014; van den Oord et al., 2016].

Machine translation is to realize a translation system by using a computer task, and the machine translation task has been tackled for decades; example-based machine translation model [Nagao, 1984], statistical machine translation models [Brown et al., 1993; Koehn et al., 2003; Chiang, 2007]. The goal of the machine translation task is to generate a translated sentence of an input sentence, which was modeled as a conditional language model. In the statistical machine translation approaches divided the model components and optimized each component so that the whole model can achieve the better performance. The recent neural network-based approach called neural machine translation (NMT) model is directly modeled and trained as a one fully connected network, and this neural network-based machine translation models have outperformed the statistical machine translation models and achieved the state-of-the art performance in several language translation tasks.

The NMT models are based on the idea of the *Encoder-Decoder* model [Cho et al., 2014b; Sutskever et al., 2014] which converts the input data into a vector space and generates the output data from the vector space. Here, every word, phrase, and sentence are represented as a vector. There have been many studies in building a vector space for words [Mikolov et al., 2013a; Mikolov et al., 2013c], phrases [Socher et al., 2010], sentences [Pennington et al., 2014; Socher et al., 2013], and paragraphs [Dai et al., 2014]. In the NMT models, the sentences as input data and output data are considered as a sequence of the word units. Recurrent neural network [Elman, 1990] is a time-series neural network and is employed as *Encoder* and *Decoder*. Thus, the early NMT models are based on sequence-to-sequence learning. Any syntactic information of the language data were not utilized explicitly, while it is well known that syntactic information are effective in statistical machine translation models [Yamada and Knight, 2001; Liu et al., 2006].

A sentence has a structure inherited in a language, and a phrase structure builds the sentences in a bottom-up fashion. The syntactic structure can be represented as a tree-structured data. Following the structured data, Socher et al. (2011) built a recursive neural network model which computes the phrase vectors. The recursive neural network model has been reported to have strength in long dependency relations in a sentence in comparison to the sequence-based recurrent neural networks [Li et al., 2015]. Sennrich and Haddow (2016) have improved the NMT performance by adding the syntactic labels as additional input features such as part-of-speech tags, dependency labels, lemmatized tags and so on. There were no previous studies which incorporated the syntactic structure of the language into the existing NMT models. Hence, this thesis aims to build the novel NMT models which incorporates the syntactic structures, to experiment the proposed model on the real data and verify the effectiveness of the proposed model, and to investigate the improvement of employing the syntactic structures in a translation task.

1.2 Proposal of the Thesis

In this thesis, we have proposed novel methods to utilize syntactic information inherited in a language and to develop the syntax-based neural machine translation models, where all of the natural language processing data i.e. words and sentence are represented in a vector space. We especially concentrate on two types of syntactic structures; 1) phrase structures and 2) universal dependency tree structures. We have proposed the syntax-based model to incorporate the syntactic structures into the NMT models.

Tree-to-sequence NMT model

We have proposed a tree-based encoder which computes not only the sequential vectors at

each time step but also the phrase vectors following the phrase structures of an input sentence. We employ an external parsing tool to estimate the phrase structures of each sentence on the training corpora. Since all of the sentences can not always be parsed completely, we model the tree-based encoder which incorporates the syntactic structures if any and otherwise works as the original sequential encoder. Our proposed model is an expanded model of the existing sequence-to-sequence NMT model. Moreover, we aim to let the model make use of the phrase vectors in the source side and softly align a target word with source words as well as source phrase by employing attention mechanism in our proposed model. We also develop a speed up technique to reduce the computational cost of the NMT models.

Sequence-to-Tree NMT model

We have proposed a syntax-based decoder which generates a translated sentence and parse the sentence simultaneously. Here, we focus on a dependency relations between words in a target sentence and extract the syntactic structure by using a universal dependency parser. Universal dependencies[1] is a framework for cross-linguistically consistent grammatical annotations over 60 languages[2]. The current NMT decoder is based on recurrent neural network and lacks of long dependent relations when training. We aim to build a new NMT model which jointly learn to translate and parse. At test time, we let our proposed model translate a sentence and generate its parsed tree optionally.

1.3 Contribution of the Thesis

We state four main contributions of this thesis as follows:

Exploration of how to model syntactic structures in NMT

Early NMT models have been developed as a sequence-to-sequence model. Since there was no previous work which utilized syntactic structures in a language, we can say that it is a challenging task to explore the ways of modeling the syntactic structures. This thesis mainly studies these two types of syntactic structures: 1) phrase structure in a source language and 2) dependency tree in a target language.

Confirmation of the effectiveness of using phrase structure in a source language

We extended the existing sequence-to-sequence NMT models to utilize the phrase structures in a source language and confirmed that our proposed model achieved the higher accuracy

[1] http://universaldependencies.org/

[2] reported in 2017

than the existing neural machine translation models. Our proposed method is the first baby step of the syntactic neural machine translation approaches; therefore this research contributes to boost the syntactic approaches in the neural machine translation research area. Besides, the proposed ideas can be easily applied for the other natural language processing tasks such as summarization. We also employed an efficient sampling-based method and succeeded in saving the training time while keeping the model performance higher.

Examination of the character-based decoding in tree-to-sequence NMT model

We have proposed a tree-to-sequence model which employs source-side phrase structures, while the model generates a translated sentence word by word. We applied a character-based decoding method and reported the different trends and characteristics of the word-based decoding method and the character-based decoding method.

Development of a jointly learning method to parse and translate

When we try to incorporate syntactic structures into the NMT models, the decoding process is known to be the more different to control than the encoding process. It was considered as a difficult task to let NMT model predict a translation and its parsed tree simultaneously. Our proposed method is to jointly learn to parse and translate in one model. We focused on dependency tree structure in the target side and trained the model by using the dependency-parsed trees as the auxiliary information in addition to the translated sentences. Developing the methods to simultaneously generate the translations and their parsed tree enables us to select more grammatical sentences and to analyze the generated sentences. Our proposed approach can be applied widely because it can handle the phrase structure or any other similar structured data as a syntactic structure.

1.4 Composition and Contents of the Thesis

This thesis consists of seven main chapters as follows:

Chapter 2 Neural Networks

Neural network-based models have achieved many successes in variety of natural language tasks such as parsing task and machine translation task. In this chapter, we explain the fundamentals of neural networks and introduce the basic ideas when applying the neural network-based models to the natural language processing tasks.

Chapter 3 Neural Machine Translation

Machine translation is to build a model to translate a sentence in a language into another

language while keeping the semantics. There had been many studies on statistical machine translation models for decades while the recent studies on the NMT models rapidly progressed and achieved the state-of-the-art performance in many language pairs. This chapter introduces the basic ideas, the learning methods of NMT model. We also explain the automatic evaluation metrics for machine translation models.

Chapter 4 Tree-to-Sequence Neural Machine Translation

In this chapter, we have proposed the syntax-based neural machine translation model which utilize the syntactic structure in the source language. Most of the existing NMT models focus on the conversion of sequential data and do not directly use syntactic information in both of source and target sides. 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. We also proposed a sampling method to reduce the high computational costs of learning a neural machine translation model.

Chapter 5 Character-based Decoding in Tree-to-Sequence Model

Most of the NMT models let the decoder generate a sentence word by word, while the word-based decoder is known to have the problems of the low coverage of the vocabulary. When building a character-based vocabulary from the training corpora, we resolve the vocabulary coverage problems easily. We apply this character-based approach to our proposed tree-to-sequence NMT model, and we report each trend of the word-based decoder and the character-based decoder.

Chapter 6 Sequence-to-Tree Neural Machine Translation

There has been relatively little attention to incorporating linguistic prior to neural machine translation. Much of the previous work was further constrained to considering linguistic prior on the source side. In this chapter, we propose a hybrid model, called NMT+RNNG, that learns to parse and translate by combining the recurrent neural network grammar into the attention-based neural machine translation. Our approach encourages the neural machine translation model to incorporate linguistic prior during training, and lets it translate on its own afterward. Extensive experiments with four language pairs show the effectiveness of the proposed NMT+RNNG.

Chapter 7 Conclusion

In this chapter, we state the conclusion of this thesis and mention future work in NMT area.

Chapter 2

Neural Networks

This chapter introduces the fundamentals of neural networks and the basic ideas to apply the neural networks to natural language tasks. We give a sentence Neural networks is one of the machine learning algorithms, and there have been many studies so far. Here, we explain the fundamentals of the neural networks by illustrating multi-layered perceptron and also introduce the application to the NLP tasks such as a sentiment classification task.

2.1 Multi-layered perceptron

We explain a multi-layered perceptron by giving sentiment classification task as an example. Sentiment classification task is to classify the text data into two categories of "*positivie*" and "*negative*", so this task is a binary classification task. Now, we have the training data which is composed of the n pairs of a sentence and its sentiment label ($(\boldsymbol{x}^i, y^i) \in \mathcal{D}_{train} (i = 1, 2, \cdots, N)$). Each sentence has T_i words defined as:

$$\boldsymbol{x}^i = (x_1^i, x_2^i, \cdots, x_{T_i}^i). \tag{2.1}$$

Each sentiment label is represented as:

$$y_j^i = \begin{cases} 1 & (j = positive) \\ 0 & (j = negative) \end{cases} \tag{2.2}$$

In this sentiment classification task, we train the multi-layered perceptron model as a model to predict the correct label y^i as an output when feeding the text $\boldsymbol{x}^i$ as an input. Figure 2.1 shows an outline image of the sentiment classification task.

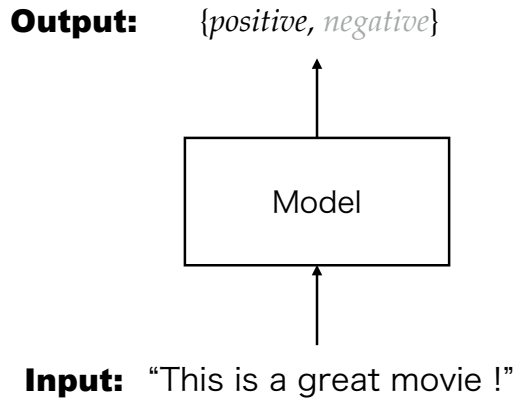


Figure 2.1: Sentiment classification task.

The multi-layered perceptron is an algorithm of supervised learning and basically compose of three types of networks, that is to say, an input layer, n-layered hidden layer, and an output layer. Each layer is defined as a vector. Assuming that we have 3-layered perceptron here ($n = 1$), the input layer is defined as the d_V-dimensional vector $\boldsymbol{i} \in \mathbb{R}^{d_V \times 1}$ as follows:

$$\boldsymbol{i} = (v_1, v_2, \cdots, v_{d_V}). \tag{2.3}$$

The input vector of $\boldsymbol{i}$ represents the prefixed d_V-length vector as we convert the input data of $\boldsymbol{x}^i$ to the predefined features (e.g. 2-gram feature) beforehand. The other layers of the hidden layer and the output layer are similarly represented as follows:

$$\boldsymbol{h} = (h_1, h_2, \cdots, h_{d_H}), \tag{2.4}$$

$$\boldsymbol{o} = (o_1, o_2, \cdots, o_{d_O}), \tag{2.5}$$

where $\boldsymbol{h} \in \mathbb{R}^{d_H \times 1}$ and $\boldsymbol{o} \in \mathbb{R}^{d_o \times 1}$ are the d_V-dimensional vector and the d_O-dimensional vector, respectively.

The vector in the hidden layer is calculated from the vector in the input layer as follows:

$$\boldsymbol{h} = f_h(\boldsymbol{W}_h \boldsymbol{i} + \boldsymbol{b}_h), \tag{2.6}$$

where $\boldsymbol{W}_h \in \mathbb{R}^{d_H \times d_V}$ and $\boldsymbol{b}_h \in \mathbb{R}^{d_H \times 1}$ are a weight matrix and a bias vector. The function f_h is a non-linear function and applied to each element of the vectors. In a similar manner, the vector in the output layer is computed from the vector in the hidden layer as follows:

$$\boldsymbol{o} = f_o(\mathbf{W}_o \boldsymbol{h} + \boldsymbol{b}_o), \tag{2.7}$$

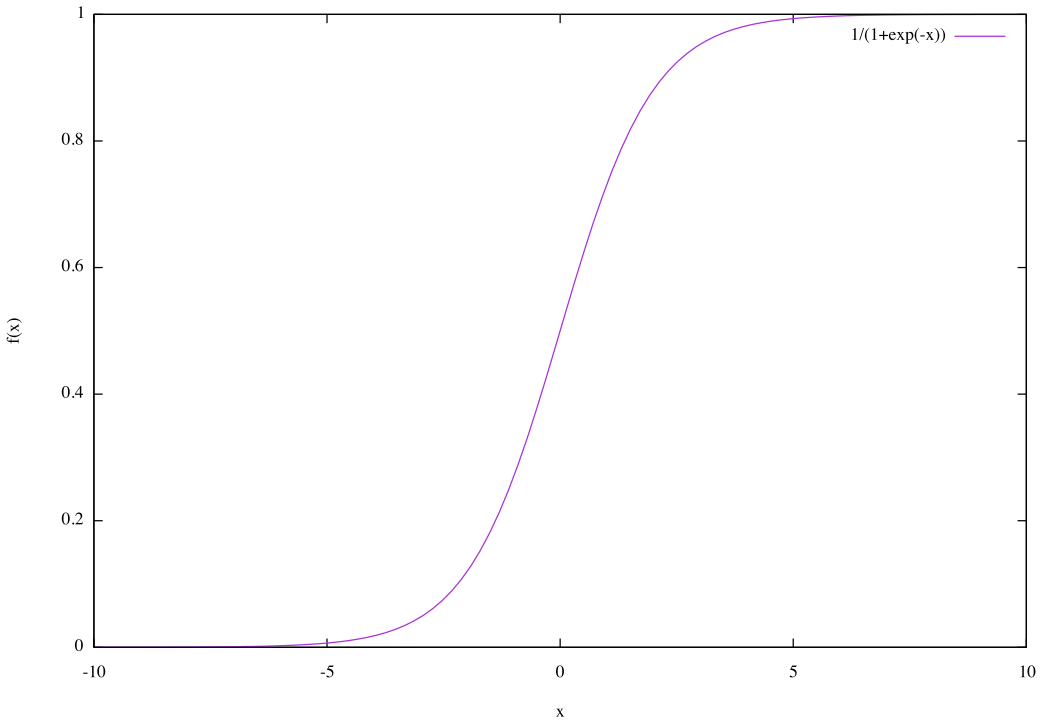


Figure 2.2: The plot of the sigmoid function (Equation (2.8)).

where $\boldsymbol{W}_h \in \mathbb{R}^{d_O \times d_H}$ and $\boldsymbol{b}_h \in \mathbb{R}^{d_O \times 1}$ are a weight matrix and a bias vector. The function f_o is a non-linear function.

One of the representative non-linear functions is the sigmoid function, which is computed as:

$$f(x) = \sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}}. \tag{2.8}$$

Figure 2.2 shows the range of the output $f(x) \in (0, 1)$ when we apply the sigmoid function to the input x. The differentiated sigmoid function with respect to x is expressed with the sigmoid function itself as follows:

$$\frac{\partial f(x)}{\partial x} = f(x)(1 - f(x)). \tag{2.9}$$

In the output layer, we predict a label in a classification task and compute the probability of the sentiment label $p(y_j^i|\boldsymbol{x}^i)$ by employing the softmax function defined as:

$$p(y_j^i|\boldsymbol{x}^i) = f(z_j) = softmax(z_j) = \frac{\exp(z_j)}{\sum_{k=1}^{d_O} \exp(z_k)}, \tag{2.10}$$

where z_j denotes the j-th label. The softmax function normalized Note that $d_O = 2$ in the sentiment classification task. Here, we assume each function in Equation (2.6) and Equation (2.7) as follows:

$$f_h = \sigma(\boldsymbol{x}), \tag{2.11}$$

$$f_o = softmax(\boldsymbol{x}). \tag{2.12}$$

Both of the functions are applied to every elements in the input vectors of $\boldsymbol{x}$. Consequently, the 3-layered networks defined above consists of these model parameters $\boldsymbol{\theta}$ such as:

$$\boldsymbol{\theta} = (\boldsymbol{W}_h, \boldsymbol{W}_o, \boldsymbol{b}_h, \boldsymbol{b}_o). \tag{2.13}$$

The purpose of supervised learning here is to learn the optimal model parameters $\boldsymbol{\theta}$ which output the correct label of y_j^i when feeding the input datum of $\boldsymbol{x}^i$. We define the loss of the model with respect to i-th data pair of $(\boldsymbol{x}^i, y^i)$ by using the cross entropy loss function defined as:

$$C = -\sum_j \log(softmax(z_j)) y_j^i. \tag{2.14}$$

Since we already assumed that there are N pairs of training data such as $(\boldsymbol{x}^i, y^i) \in \mathcal{D}_{train} (i = 1, 2, \cdots, N)$, we define the objective function by averaging the cross entropy losses up with respect to the i-th training datum.

$$Obj(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_j \log(softmax(z_j)) y_j^i \tag{2.15}$$

When training the multi-layered perceptron in the sentiment classification task, we aim to obtain the model parameters $\boldsymbol{\theta}$ which minimize the objective function in Equation (2.15).

2.2 Stochastic Gradient Decent Method

When training the model, we optimize the parameters by updating so that we can minimize the predefined objective function. We introduce the stochastic gradient descent method here, and the algorithm of the stochastic gradient descent is described as follows:

1. Sample n data randomly from the training data.
2. Calculate the averaged of the loss function of the data

$$\overline{Obj(\boldsymbol{\theta})} = \frac{1}{n} \sum_{i=1}^{n} Obj_i(\boldsymbol{\theta}) \tag{2.16}$$

3. Compute the derivatives of each model parameter

4. Update the parameter theta following the Equation (2.17).

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{\partial \overline{Obj(\boldsymbol{\theta})}}{\partial \boldsymbol{\theta}}, \tag{2.17}$$

where α is a learning rate and a hyper parameter. When training the data, we generally do mini-batch learning to update the model parameters by summing the predefined number of training data up at once.

Chapter 3

Neural Machine Translation

This chapter explain the basic elements of a machine translation task and neural machine translation (NMT) models. The machine translation task is to translate a sentence written in one language into a sentence written in another language, and the machine translation model is trained on training data called a parallel corpus. The NMT model has been proposed as an end-to-end neural network-based machine translation model which is based on the idea of an *Encoder-Decoder* model. Moreover, the performance of the NMT models are improved by introducing the idea of *attention mechanism*. The Encoder-Decoder model with attention mechanism has been recently referred as a neural machine translation model.

3.1 Language model based on recurrent neural network

In natural language processing task, a sentence $\boldsymbol{x}$ in a language is often modeled as a language model to predict the next word as follows:

$$p(\boldsymbol{x}) = \sum_i p(x_i|\boldsymbol{x}_{<i}). \tag{3.1}$$

Bengio et al. (2003) have proposed a neural network-based language model, and [Mikolov et al., 2010] employed recurrent neural network (RNN) [Elman, 1990] for modeling a language model. RNN is a type of neural network for time-series data which computes each time step vector. We calculate the i-th hidden unit $\boldsymbol{h}_i \in \mathbb{R}^{d\times 1}$ given the i-th input x_i and the previous hidden unit $\boldsymbol{h}_{i-1} \in \mathbb{R}^{d\times 1}$ as follows:

$$\boldsymbol{h}_i \quad = \quad f(V_x(x_i), \boldsymbol{h}_{i-1}), \tag{3.2}$$

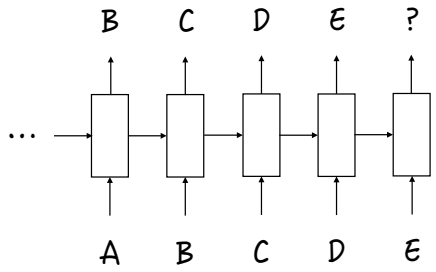


Figure 3.1: An illustration of RNN-based language model.

where f is a non-linear function and $V_x(x_[i)$ is a word embedding vector of x_i. Given the previous predicted word x_{i-1} and the i-th hidden unit of the recurrent neural network, the conditional probability that the i-th word is generated by employing a softmax function:

$$p(x_i|\boldsymbol{x}_{<i}) \quad = \quad softmax(\boldsymbol{W}[\boldsymbol{h}_{i-1}; V_x(x_{i-1})] + \boldsymbol{b}), \tag{3.3}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are a matrix and a bias vector. $[\boldsymbol{h}_{i-1}; V_x(x_{i-1})]$ is a concatenation of both vectors. Figure 3.1 shows an illustration of the RNN-based language model.

3.2 Encoder-Decoder Model as a Conditional Language 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. Figure 3.2 shows an illustration of the RNN-based sequence-to-sequence learning model.

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 $\boldsymbol{h}_{i-1} \in \mathbb{R}^{d\times 1}$,

$$\boldsymbol{h}_i \quad = \quad f_{enc}(V_x(x_i), \boldsymbol{h}_{i-1}), \tag{3.4}$$

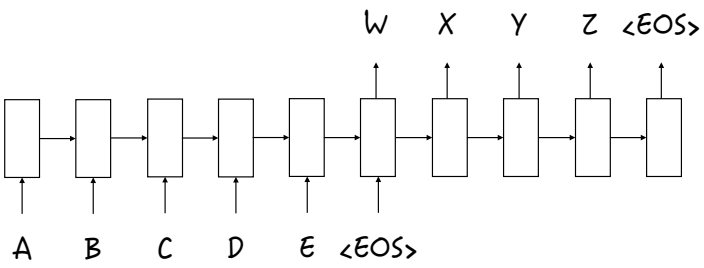


Figure 3.2: An illustration of sequence-to-sequence model.

where f_{enc} is a non-linear function, $V_x(x_i)$ returns the word embedding vector of x_i, 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 $\boldsymbol{s}_1 \in \mathbb{R}^{d \times 1}$ is initialized with the input sentence vector.

$$\boldsymbol{s}_1 = \boldsymbol{h}_n. \tag{3.5}$$

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:

$$p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) \quad = \quad g(\boldsymbol{s}_j), \tag{3.6}$$

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:

$$\boldsymbol{s}_j \quad = \quad f_{dec}(V_y(y_{j-1}), \boldsymbol{s}_{j-1}), \tag{3.7}$$

where V_y returns the word embedding vector of y_{j-1}.

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 $\boldsymbol{c}_t \in \mathbb{R}^{d\times 1}$,

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{U}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}),\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{U}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}),\\
\tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{W}^{(\tilde{c})}\boldsymbol{x}_t + \boldsymbol{U}^{(\tilde{c})}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(\tilde{c})}),\\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1},\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t),
\end{aligned} \tag{3.8}$$

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.

3.3 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:

$$\boldsymbol{\alpha}_j(i) = \frac{\exp(\boldsymbol{h}_i \cdot \boldsymbol{s}_j)}{\sum_{k=1}^{n}\exp(\boldsymbol{h}_k \cdot \boldsymbol{s}_j)}, \tag{3.9}$$

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

$$\boldsymbol{d}_j = \sum_{i=1}^{n}\boldsymbol{\alpha}_j(i)\boldsymbol{h}_i. \tag{3.10}$$

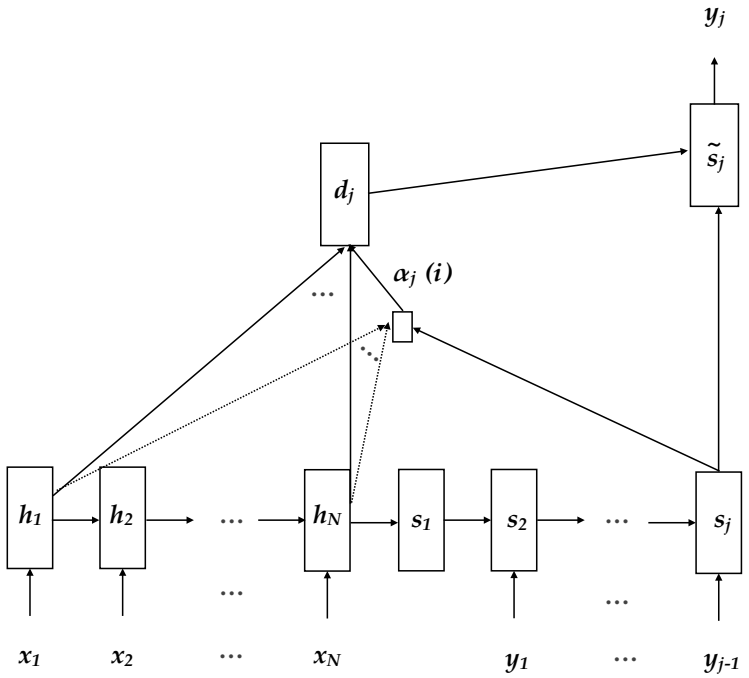


Figure 3.3: Attentional Encoder-Decoder model.

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 $\tilde{\boldsymbol{s}}_j$:

$$\tilde{\boldsymbol{s}}_j = \tanh(\boldsymbol{W}_d[\boldsymbol{s}_j;\ \boldsymbol{d}_j] + \boldsymbol{b}_d), \tag{3.11}$$

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:

$$p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_s\tilde{\boldsymbol{s}}_j + \boldsymbol{b}_s), \tag{3.12}$$

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 3.3 shows an example of the NMT model with the attention mechanism.

3.4 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:

$$J(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \log p(\boldsymbol{y}|\boldsymbol{x}), \tag{3.13}$$

where $\mathcal{D}$ denotes a set of parallel sentence pairs. The model parameters $\boldsymbol{\theta}$ are learned through Stochastic Gradient Descent described in Subsection 2.2.

3.5 Evaluation Metrics

It is also important how to evaluate a machine translation system we construct. There are two types of evaluation manners; 1) Human evaluation and 2) Automatic evaluation. In the former case, the people who are familiar to both of languages evaluate the translations obtained by the machine translation systems. Human evaluation is well known as a highly reliable evaluation method and actually employed at several machine translation conference or workshops such as the first conference on Machine Translation'16 [1], it however takes much time and costs high to check the qualities of the translation by human. In the latter case, we automatically compute the translation quality based on the predefined indicator, but the automatic evaluation metrics lacks of reliability in comparison to human evaluation. There are still open research questions of how to handle semantics of the sentences and how to evaluate them objectively.

There are several automatic evaluation metics proposed already such as BLEU[Papineni et al., 2002], RIBES[Isozaki et al., 2010], and TER[Snover et al., 2006]. Given golden translation called reference and translation obtained by a machine translation system, we calculate the BLEU score based on the n-gram word precisions between both of the corpus. The BLEU score is calculated as follows:

$$\prod_{n=1}^{4} p_n^{\frac{1}{4}} \times \min(\exp(1 - \frac{\sum_{s=1}^{S} |\mathbf{r}_s^*|}{\sum_{s=1}^{S} |\mathbf{e}_s^*|}), 1) \tag{3.14}$$

where $\mathbf{e}_s^*$ are the translations obtained by a machine translation system and $\{\{\mathbf{r}_s^i\}_{i=1}^R\}_{s=1}^S$ $(R \leq 1)$ are the references. We calculate the geometry mean of the precision of n-gram $(1 \leq n \leq 4)$, and p_n denotes the precision of n-gram words.

[1] http://www.statmt.org/wmt16/

Chapter 4

Tree-to-Sequence Neural Machine Translation

This chapter has proposed the first syntax-based neural machine translation. We focus on phrase structures inherited in a source sentence and incorporate the phrase structure into the existing neural machine translation model This chapter was already published as a conference paper as follows:

Chapter 4 Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. Tree-to-Sequence Attentional Neural Machine Translation. *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 823–833, Berlin, Germany, August 2016.

4.1 Introduction

Machine Translation 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

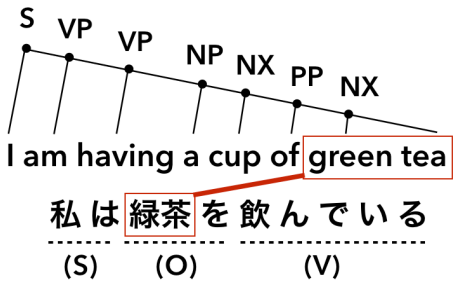


Figure 4.1: Alignment between an English phrase and a Japanese word.

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 4.1 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 chapter, we propose a novel attention-based NMT model to take advantage of syntactic information. 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.

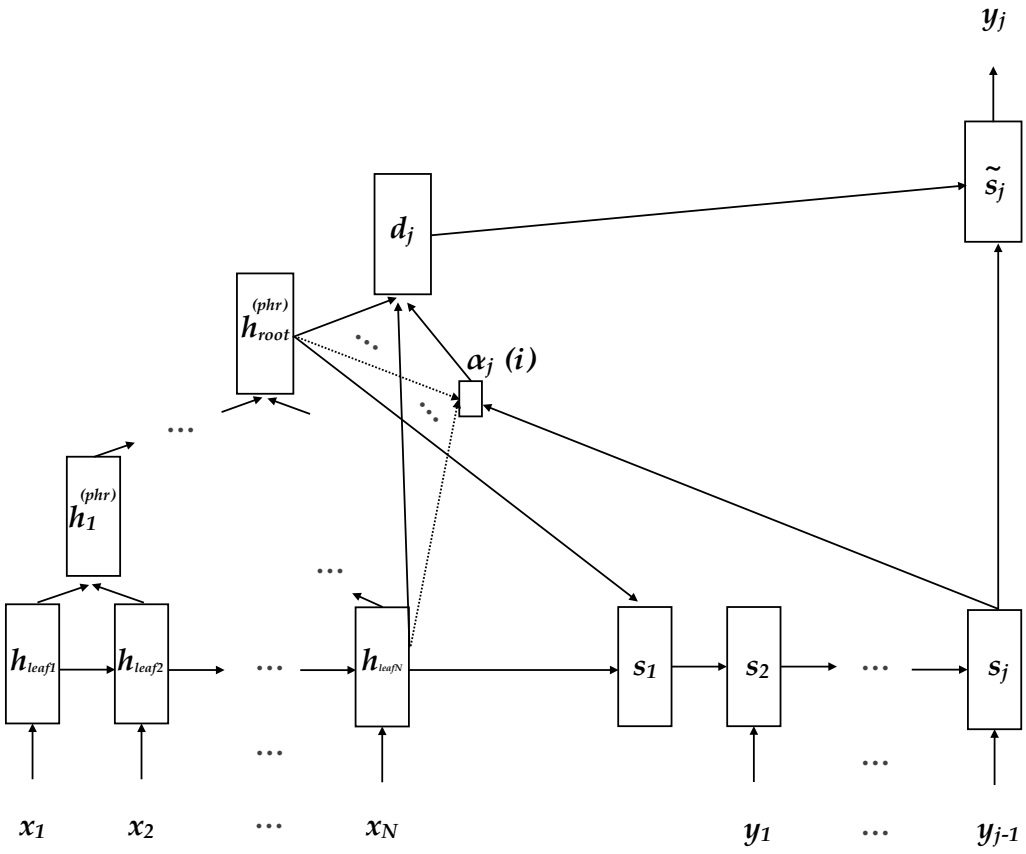


Figure 4.2: Proposed model: Tree-to-sequence attention-based NMT model.

4.2 Attention-based Tree-to-Sequence Model

4.2.1 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 4.2 shows our proposed model, which we call a *tree-to-sequence attention-based NMT model*.

In Head-driven Phrase Structure Grammar (HPSG) [Sag et al., 2003], a sentence is composed

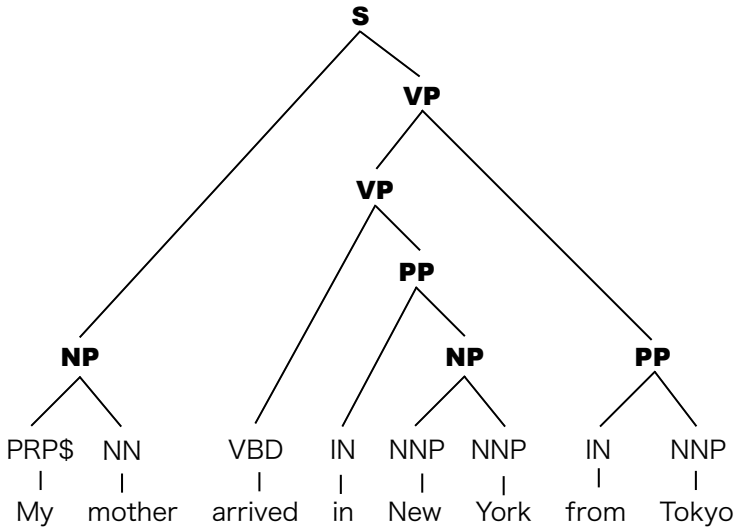


Figure 4.3: An example of phrase structure of the sentence "My mother arrived in New York from Tokyo".

of multiple phrase units and represented as a binary tree as shown in Figure 4.4. Here, the leaf nodes show the words in a sentence and the part-of-speech tags corresponding to the words and the non-leaf nodes denote the phrase nodes with the corresponding phrase category labels. 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:

$$\boldsymbol{h}_k^{(phr)} = f_{tree}(\boldsymbol{h}_k^l, \boldsymbol{h}_k^r), \tag{4.1}$$

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 3.2. 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

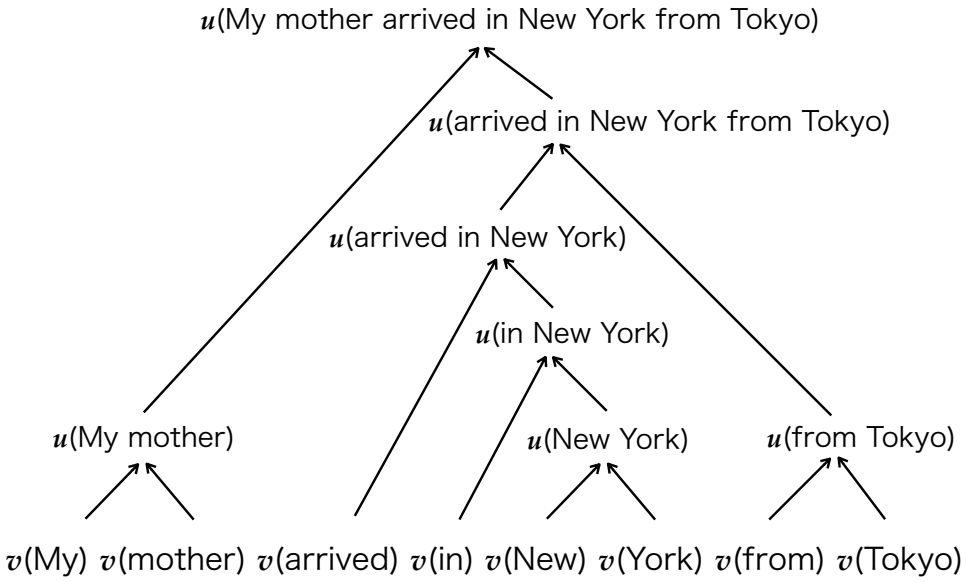


Figure 4.4: An illustration of the way to calculate a phrase vector $\boldsymbol{u}$ in each node from the word vectors $\boldsymbol{v}$ in the sentence "My mother arrived in New York from Tokyo".

cell $\boldsymbol{c}_k^{(phr)} \in \mathbb{R}^{d\times 1}$ for the k-th parent node are calculated as follows:

$$
\begin{aligned}
\boldsymbol{i}_k &= \sigma(\boldsymbol{U}_l^{(i)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(i)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_k^l &= \sigma(\boldsymbol{U}_l^{(f_l)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_l)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(f_l)}),\\
\boldsymbol{f}_k^r &= \sigma(\boldsymbol{U}_l^{(f_r)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_r)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(f_r)}),\\
\boldsymbol{o}_k &= \sigma(\boldsymbol{U}_l^{(o)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(o)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(o)}),\\
\tilde{\boldsymbol{c}}_k &= \tanh(\boldsymbol{U}_l^{(\tilde{c})}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(\tilde{c})}\boldsymbol{h}_k^r + \boldsymbol{b}^{(\tilde{c})}),\\
\boldsymbol{c}_k^{(phr)} &= \boldsymbol{i}_k \odot \tilde{\boldsymbol{c}}_k + \boldsymbol{f}_k^l \odot \boldsymbol{c}_k^l + \boldsymbol{f}_k^r \odot \boldsymbol{c}_k^r,\\
\boldsymbol{h}_k^{(phr)} &= \boldsymbol{o}_k \odot \tanh(\boldsymbol{c}_k^{(phr)}),
\end{aligned}
\tag{4.2}
$$

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 LSTM units 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.

4.2.2 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 4.2, 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:

$$\boldsymbol{s}_1 = g_{tree}(\boldsymbol{h}_n, \boldsymbol{h}_{root}^{(phr)}), \tag{4.3}$$

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.

4.2.3 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 $\boldsymbol{\alpha}_j(i)$:

$$\boldsymbol{d}_j = \sum_{i=1}^{n} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i + \sum_{i=n+1}^{2n-1} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i^{(phr)}. \tag{4.4}$$

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 (3.11).

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 $\boldsymbol{s}_j$,

$$\boldsymbol{s}_j = f_{dec}(y_{j-1}, [\boldsymbol{s}_{j-1};\ \tilde{\boldsymbol{s}}_{j-1}]), \tag{4.5}$$

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.

4.2.4 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 (3.12), 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 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 (**??**) 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., 2013b]. 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 RNN language models. 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.

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

Table 4.1: Dataset in ASPEC corpus.

4.3 Experiments

4.3.1 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 5.2 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.

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 4.2 shows the details of each training dataset and its corresponding vocabulary size.

4.3.2 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),

[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

	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 4.2: Training dataset and the vocabulary sizes.

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 stochastic gradient descent with the mini-batch size of 128. The initial learning rate of stochastic gradient descent 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 attention-based 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.

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 $d = 512$.

4.3.3 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:

$$score(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{m} \log p(y_j | \boldsymbol{y}_{<j}, \boldsymbol{x}). \tag{4.6}$$

[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

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:

$$score(\boldsymbol{x}, \boldsymbol{y}) = \quad L_{\boldsymbol{x},\boldsymbol{y}} + \sum_{j=1}^{m} \log p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}), \tag{4.7}$$

$$L_{\boldsymbol{x},\boldsymbol{y}} = \quad \log p(len(\boldsymbol{y})|len(\boldsymbol{x})), \tag{4.8}$$

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.

4.3.4 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).

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

4.4 Results and Discussion

4.4.1 Small Training Dataset

Table 4.3 shows the perplexity, BLEU, RIBES, and the training time on the development data with the attention-based NMT 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 4.3, 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 ($K = 500$ and 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 attention-based NMT 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 4.3. 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 the negative samples multiple times at each training step.

	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
Attention-based NMT model [Luong et al., 2015a]	500	21.6	70.7	18.5	45
+ reverse input	500	22.6	69.8	17.7	45
Attention-based NMT model [Luong et al., 2015a]	2000	23.1	71.5	19.4	60
+ reverse input	2000	26.1	69.5	17.5	60

Table 4.3: 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	Brevity Penalty
Simple Beam Search	6	72.3	20.0	0.901
	20	72.3	19.5	0.851
Proposed Beam Search	20	**72.6**	**20.5**	0.917

Table 4.4: Effects of the Beam Search on the development data.

	RIBES	BLEU
Without sequential LSTM units	69.4	19.5
With sequential LSTM units	**72.3**	**20.0**

Table 4.5: Effects of the sequential LSTM units in our proposed tree-based encoder on the development data.

Effects of the proposed beam search Table 4.4 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 LSTM units at the leaf nodes in our proposed tree-based encoder. Table 4.5 shows the result on the development data of our proposed encoder and that of an attentional tree-based encoder without sequential LSTM units with BlackOut ($K = 2000$).[7] 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. Figure 4.5 shows an illustration of the proposed model without sequential LSTM units.

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

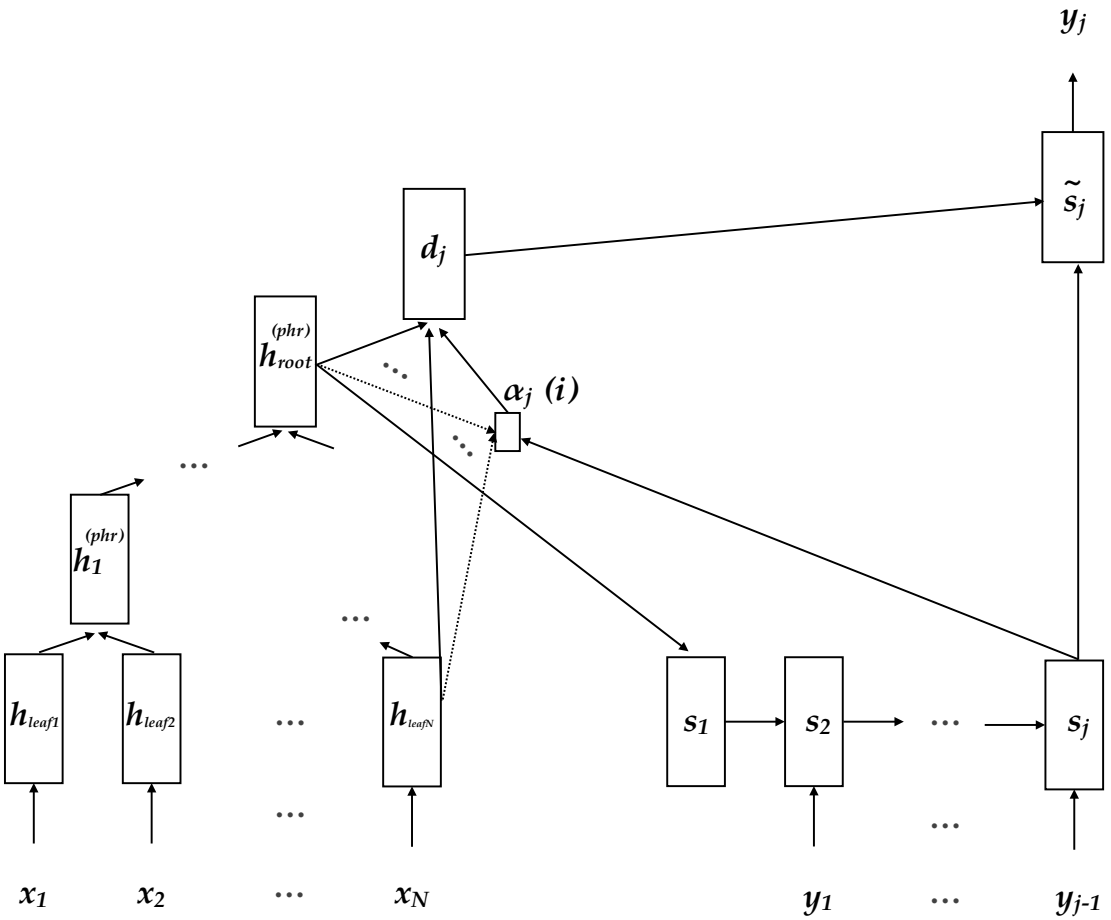


Figure 4.5: Proposed tree-to-sequence model without sequential LSTM units.

4.4.2 Large Training Dataset

Table 4.6 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

[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).

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 attention-based NMT model [Bahdanau et al., 2015] with a bi-directional LSTM encoder, and uses 1024-dimensional hidden units and 1000-dimensional word embeddings. The model of Lee et al. (2015) is also an attention-based NMT 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 attention-based NMT 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 attention-based NMT and SMT models by $+1.54$ RIBES and by $+0.74$ BLEU scores and Lee et al. (2015)'s attention-based NMT 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 attention-based NMT using a bi-directional LSTM encoder. Our proposed end-to-end NMT model compares favorably with Neubig et al. (2015).

4.4.3 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 4.6 and 4.7, an English sentence represented as a binary tree is translated into Japanese, and several attentional relations between English words or phrases and 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 4.6, 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-

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

	RIBES	BLEU
Neural Machine Translation model		
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
Attention-based NMT with LSTM units [Zhu, 2015]	79.70	32.19
+ Ensemble, *unk* replacement	80.27	34.19
+ System combination,3 pre-reordered ensembles	80.91	36.21
Attention-based NMT with GRUs [Lee et al., 2015] + character-based decoding, Begin/Inside representation	81.15	35.75
Statistical Machine Translation model		
Phrase-based SMT baseline	69.19	29.80
Hierarchical Phrase-based STM baseline	74.70	32.56
Tree-to-string SMT baseline	75.80	33.44
Tree-to-string SMT model [Neubig and Duh, 2014]	79.65	36.58
+ Attention-based NMT Rerank [Neubig et al., 2015]	81.38	38.17

Table 4.6: Evaluation results on the test data.

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 4.1.

In Figure 4.7, 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 "Ｓ ｉ 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

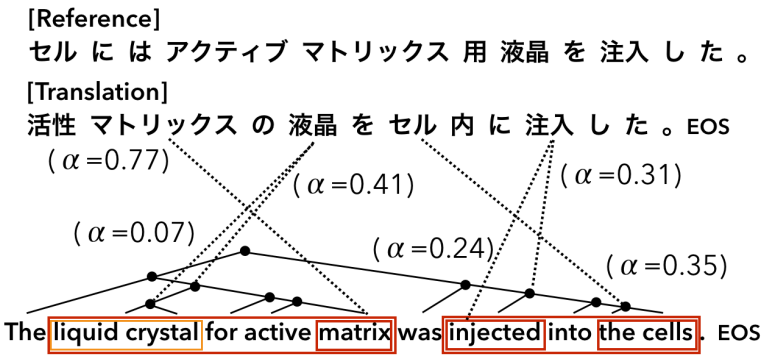


Figure 4.6: Translation example of a short sentence and the attentional relations by our proposed model.

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 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].

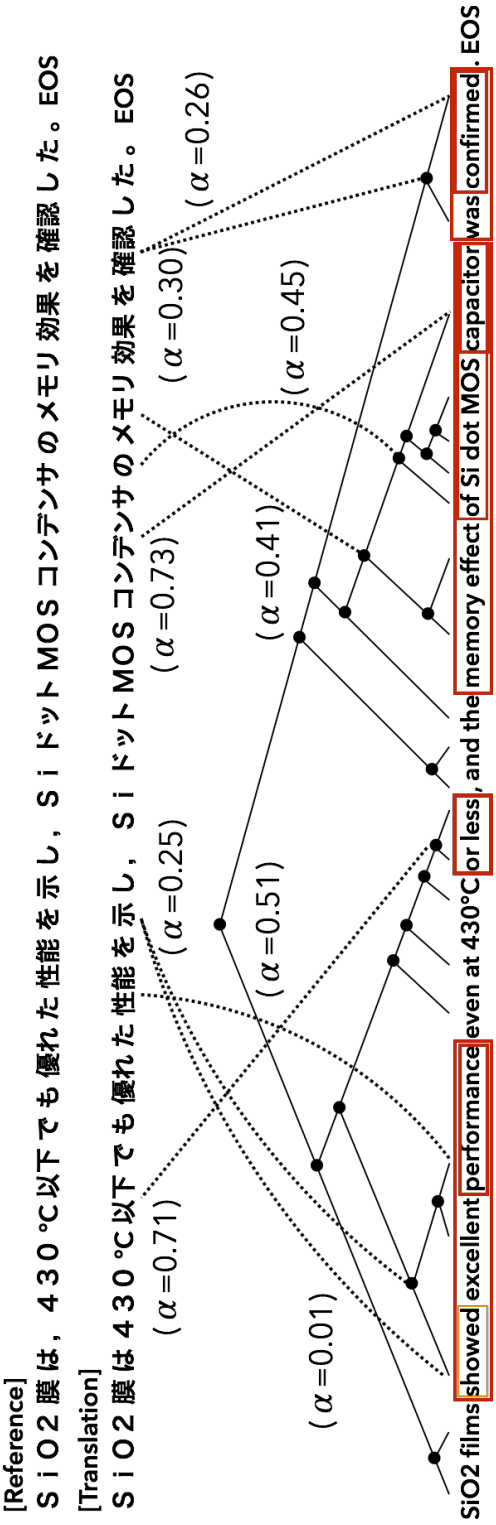


Figure 4.7: Translation example of a long sentence and the attentional relations by our proposed model.

4.5 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 LSTM units [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 attention-based NMT models. Subsequently, several attention-based NMT models have been proposed [Cheng et al., 2016; Cohn et al., 2016]; however, each model is based on the existing sequential attention-based models and does not focus on a syntactic structure of languages.

4.6 Conclusion

In this paper, we propose a novel syntactic approach that extends attention-based 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 attention-based NMT model.

Chapter 5

Character-based Decoding in Tree-to-Sequence Model

This chapter applied the character-based decoding method in the tree-to-sequence attention-based neural machine translation model. We also explored the effectiveness of the phrase category label information into tree-based encoder. This chapter was already published as a workshop paper as follows:

Chapter 5 Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. Character-based Decoding in Tree-to-Sequence Attention-based Neural Machine Translation. *In Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pp. 175–183, Osaka, Japan, December 2016..

5.1 Introduction

End-to-end Neural Machine Translation (NMT) models have recently achieved state-of-the-art results in several translation tasks [Luong et al., 2015a; Luong et al., 2015b]. Those NMT models are based on the idea of sequence-to-sequence learning [Sutskever et al., 2014], where both of the source and the target sentences are considered as a sequence of symbols (e.g. words or characters) and they are directly converted via a vector space. The sequence of symbols on the source side is input into a vector space, and the sequence of symbols on the target side is output from the vector space. In the end-to-end NMT models, the above architectures are embodied by a single neural network.

The optimal unit for NMT is an important research question discussed in the community. Early NMT models employ a word as a unit of the sequence [Cho et al., 2014b; Sutskever et al., 2014]. Sennrich et al. (2016) have used a Byte-Pair Encoding (BPE) method to create a sub-word level vocabulary according to the frequencies of sub-word appearance in the corpus. They successfully replaced a large word vocabulary in German and Russian with a much smaller sub-word vocabulary. They have also shown that their sub-word-based NMT model gives better translations than the word-based NMT models.

The smallest unit of a sequence of text data is a character. The character-based approach has attracted much attention in the field of NMT, because it enables an NMT model to handle all of the tokens in the corpus [Costa-jussà and Fonollosa, 2016; Chung et al., 2016]. A hybrid model of the word-based and the character-based model has also been proposed by Luong and Manning (2016). These studies reported the success and effectiveness in translating the out-of-vocabulary words.

In this chapter, we apply character-based decoding to a tree-based NMT model [Eriguchi et al., 2016]. The existing character-based models focus only on the sequence-based NMT models. The objective of this study is to analyze the results of the character-based decoding in the tree-based NMT model. We also enrich the tree-based encoder with syntactic features. Figure 5.1 shows an overview of the tree-to-sequence attention-based NMT model. We conducted the English-to-Japanese translation task on the WAT'16 dataset. The results of our character-based decoder model show that its translation accuracy is lower than that of the word-based decoder model by 1.34 BLEU scores, but the character-based decoder model needed much less time to generate a sentence.

5.2 Neural Machine Translation

End-to-end NMT models have recently outperformed phrase-based statistical machine translation (SMT) models in several languages [Luong et al., 2015a; Eriguchi et al., 2016]. Those NMT models are basically composed of two processes called an *encoder* and a *decoder*. We feed a sequence of words $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ in the source language into the encoder, and the encoder converts the input data into a vector space until the last n-th word in the sentence is input. Recurrent Neural Networks (RNNs) are used to obtain the vectors of the sequence of data in the recent NMT models. The i-th hidden state $\boldsymbol{h}_i \in \mathbb{R}^{d \times 1}$ in the RNN holds a vector computed by the current input x_i and the previous hidden state $\boldsymbol{h}_{i-1} \in \mathbb{R}^{d \times 1}$:

$$\boldsymbol{h}_i \quad = \quad RNN_{encoder}(Embed(x_i), \boldsymbol{h}_{i-1}), \tag{5.1}$$

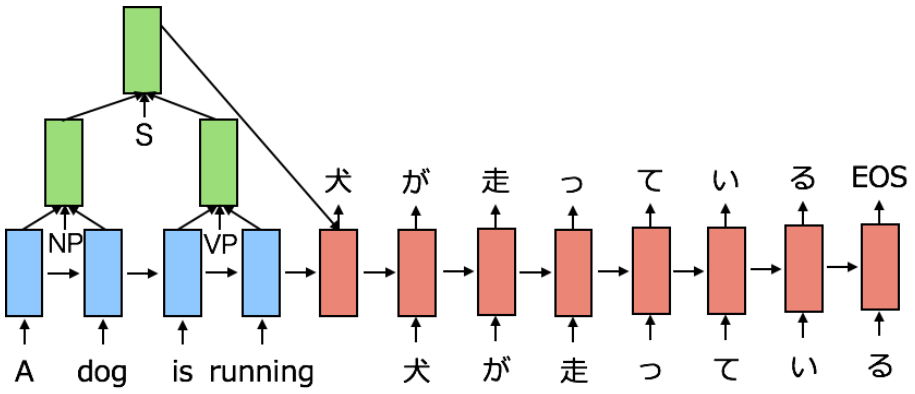


Figure 5.1: Our proposed model: tree-to-character attention-based Neural Machine Translation model.

where $Embed(x_i)$ is the word embedding vector of the i-th source word x_i. $\boldsymbol{h}_0$ is set to $\mathbf{0}$.

Another RNN is used as the decoder to obtain the vectors for predicting the words on the target side. The j-th hidden state $\boldsymbol{s}_j \in \mathbb{R}^{d\times 1}$ of the RNN is computed from the previous hidden state $\boldsymbol{s}_{j-1} \in \mathbb{R}^{d\times 1}$ and the previous output word y_{j-1} as follows:

$$\boldsymbol{s}_j \quad = \quad RNN_{decoder}(Embed(y_{j-1}), \boldsymbol{s}_{j-1}), \tag{5.2}$$

where $Embed(y_{j-1})$ is the word embedding vector of the $(j-1)$-th target word y_{j-1}. The first decoder $\boldsymbol{s}_1$ is initialized with the last hidden state of the encoder $\boldsymbol{h}_n$.

The NMT models that simply connect the above two types of RNNs cannot capture strong relations between the encoder units and the decoder unit, and they often fail to translate a long sentence. An attention mechanism has been introduced to solve the problem by creating an attention path so that the hidden states of the decoder can access each hidden state of the encoder [Bahdanau et al., 2015]. Luong et al. (2015a) have refined the calculation of the attention mechanism. In the decoder process, the attention score $\boldsymbol{\alpha}_j(i)$ is computed by the j-th hidden state of the decoder $\boldsymbol{s}_j$ and each hidden state of the encoder $\boldsymbol{h}_i$ as follows:

$$\boldsymbol{\alpha}_j(i) \quad = \quad \frac{\exp(\boldsymbol{h}_i \cdot \boldsymbol{s}_j)}{\sum_{k=1}^{n} \exp(\boldsymbol{h}_k \cdot \boldsymbol{s}_j)}, \tag{5.3}$$

where $\cdot$ represents the inner product, and its value of $\boldsymbol{h}_i \cdot \boldsymbol{s}_j$ is the similarity score between $\boldsymbol{h}_i$ and $\boldsymbol{s}_j$. The j-th context vector $\boldsymbol{d}_j \in \mathbb{R}^{d\times 1}$ are computed as the summation of the hidden states:

$$\boldsymbol{d}_j \quad = \quad \sum_{i=1}^{n} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i, \tag{5.4}$$

where each of the hidden states is weighted by $\boldsymbol{\alpha}_j(i)$. We compute the j-th final decoder $\tilde{\boldsymbol{s}}_j \in \mathbb{R}^{d\times 1}$ as follows:

$$\tilde{\boldsymbol{s}}_j = \tanh(\boldsymbol{W}_d[\boldsymbol{s}_j;\ \boldsymbol{d}_j] + \boldsymbol{b}_d), \tag{5.5}$$

where $[\boldsymbol{s}_j; \boldsymbol{d}_j] \in \mathbb{R}^{2d\times 1}$ denotes the concatenation of $\boldsymbol{s}_j$ and $\boldsymbol{d}_j$. $\boldsymbol{W}_d \in \mathbb{R}^{d\times 2d}$ is a weight matrix. $\boldsymbol{b}_d \in \mathbb{R}^{d\times 1}$ is a bias vector. The conditional probability of predicting an output is defined as follows:

$$p(y_j|\boldsymbol{x}, \boldsymbol{y}_{<j}) \quad = \quad \text{softmax}(\boldsymbol{W}_s\tilde{\boldsymbol{s}}_j + \boldsymbol{b}_s), \tag{5.6}$$

where $\boldsymbol{W}_s \in \mathbb{R}^{d\times d}$ is a matrix and $\boldsymbol{b}_s \in \mathbb{R}^{d\times 1}$ is a bias.

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

$$J(\boldsymbol{\theta}) \quad = \quad \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \log p(\boldsymbol{y}|\boldsymbol{x}), \tag{5.7}$$

where $\mathcal{D}$ denotes the set of parallel sentence pairs. When training the model, the parameters $\boldsymbol{\theta}$ are updated by stochastic gradient descent.

5.3 Tree-to-character attention-based NMT model

Our proposed model is mostly based on the tree-to-sequence attention-based NMT model described in Eriguchi et al. (2016) which has a tree-based encoder and a sequence-based decoder. They employed Long Short-Term Memory (LSTM) as the units of RNNs [Hochreiter and Schmidhuber, 1997; Gers et al., 2000]. In their proposed tree-based encoder, the phrase vectors are computed from their child states by using Tree-LSTM units [Tai et al., 2015], following the phrase structure of a sentence. We incorporate syntactic features into the tree-based encoder, and the k-th

phrase vector $\boldsymbol{h}_k^{(phr)} \in \mathbb{R}^{d\times 1}$ in our proposed model is computed as follows:

$$
\begin{aligned}
\boldsymbol{i}_k &= \sigma(\boldsymbol{U}_l^{(i)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(i)}\boldsymbol{h}_k^r + \boldsymbol{W}^{(i)}\boldsymbol{z}_k + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_k^l &= \sigma(\boldsymbol{U}_l^{(f_l)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_l)}\boldsymbol{h}_k^r + \boldsymbol{W}^{(f_l)}\boldsymbol{z}_k + \boldsymbol{b}^{(f_l)}),\\
\boldsymbol{f}_k^r &= \sigma(\boldsymbol{U}_l^{(f_r)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_r)}\boldsymbol{h}_k^r + \boldsymbol{W}^{(f_r)}\boldsymbol{z}_k + \boldsymbol{b}^{(f_r)}),\\
\boldsymbol{o}_k &= \sigma(\boldsymbol{U}_l^{(o)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(o)}\boldsymbol{h}_k^r + \boldsymbol{W}^{(o)}\boldsymbol{z}_k + \boldsymbol{b}^{(o)}),\\
\tilde{\boldsymbol{c}}_k &= \tanh(\boldsymbol{U}_l^{(\tilde{c})}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(\tilde{c})}\boldsymbol{h}_k^r + \boldsymbol{W}^{(o)}\boldsymbol{z}_k + \boldsymbol{b}^{(\tilde{c})}),\\
\boldsymbol{c}_k &= \boldsymbol{i}_k \odot \tilde{\boldsymbol{c}}_k + \boldsymbol{f}_k^l \odot \boldsymbol{c}_k^l + \boldsymbol{f}_k^r \odot \boldsymbol{c}_k^r,\\
\boldsymbol{h}_k^{(phr)} &= \boldsymbol{o}_k \odot \tanh(\boldsymbol{c}_k),
\end{aligned}
\tag{5.8}
$$

where each of $\boldsymbol{i}_k$, $\boldsymbol{o}_k$, $\tilde{\boldsymbol{c}}_k$, $\boldsymbol{c}_k$, $\boldsymbol{c}_k^l$, $\boldsymbol{c}_k^r$, $\boldsymbol{f}_k^l$, and $\boldsymbol{f}_k^r \in \mathbb{R}^{d\times 1}$ denotes an input gate, an output gate, a state for updating the memory cell, a memory cell, the memory cells of the left child node and the right child node, the forget gates for the left child and for the right child, respectively. $\boldsymbol{W}^{(\cdot)} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{U}^{(\cdot)} \in \mathbb{R}^{d\times m}$ are weight matrices, and $\boldsymbol{b}^{(\cdot)} \in \mathbb{R}^{d\times 1}$ is a bias vector. $\boldsymbol{z}_k \in \mathbb{R}^{m\times 1}$ is an embedding vector of the phrase category label of the k-th node. $\sigma(\cdot)$ denotes the logistic function. The operator $\odot$ is element-wise multiplication.

The decoder outputs characters one by one. Note that the number of characters in a language is far smaller than the vocabulary size of the words in the language. The character-based approaches thus enable us to significantly speed up the softmax computation for generating a symbol, and we can train the NMT model and generate translations much faster. Moreover, all of the raw text data are directly covered by the character units, and therefore the decoder in our proposed model requires few preprocessing steps such as segmentation and tokenization.

We also use the *input-feeding* technique [Luong et al., 2015a] to improve translation accuracy. The j-th hidden state of the decoder is computed in our proposed model as follows:

$$
\boldsymbol{s}_j = RNN_{decoder}(Embed(y_{j-1}), [\boldsymbol{s}_{j-1};\ \tilde{\boldsymbol{s}}_{j-1}]),
\tag{5.9}
$$

where $[\boldsymbol{s}_{j-1};\ \tilde{\boldsymbol{s}}_{j-1}] \in \mathbb{R}^{2d\times 1}$ denotes the concatenation of $\boldsymbol{s}_{j-1}$ and $\tilde{s}_{j-1}$.

5.4 Experiment in WAT'16 task

5.4.1 Experimental Setting

We conducted experiments for our proposed model using the 3rd Workshop of Asian Translation 2016 (WAT'16)[1] English-to-Japanese translation task [Nakazawa et al., 2016a]. The data set is the Asian Scientific Paper Excerpt Corpus (ASPEC) [Nakazawa et al., 2016b]. The data setting followed the ones in Zhu (2015) and Eriguchi et al. (2016). We collected 1.5 million pairs of training sentences from *train-1.txt* and the first half of *train-2.txt*. We removed the sentences whose lengths are greater than 50 words. In the tree-based encoder, binary trees of the source sentences were obtained by Enju [Miyao and Tsujii, 2008], which is a probabilistic HPSG parser. We used phrase category labels as the syntactic features in the proposed tree-based encoder. There are 19 types of phrase category labels given by Enju. Table 5.1In the word-based decoder model, we employed KyTea [Neubig et al., 2011] as the segmentation tool for the Japanese sentences. We performed the preprocessing steps of the data as recommended in WAT'16.[2] Table 5.2 and Table 5.3 show the details of the final dataset and the vocabulary sizes in our experiments. Each vocabulary includes the words and the characters whose frequencies exceed five or two in the training data, respectively. The out-of-vocabulary words are mapped into a special token i.e. "UNK". As a result, the vocabulary size of the characters in Japanese is about 22 times smaller than that of the words.

NMT models are often trained on a limited vocabulary, because the high computational cost of the softmax layer for target word generation is usually the bottleneck when training an NMT model. In the word-based models, we use the BlackOut sampling method [Ji et al., 2016] to approximately compute the softmax layer. The parameter setting of BlackOut follows Eriguchi et al. (2016). In the character-based models, we use the original softmax in the softmax layer. All of the models are trained on CPUs.[3] We employed multi-threading programming to update the parameters in a mini-batch in parallel. The training times of the single word-based model and the single character-based model were about 11 days and 7 days, respectively.

We set the dimension size of the hidden states to 512 in both of the LSTMs and the Tree LSTMs. The dimension size of embedding vectors is set to 512 for the words and to 256 for the characters. In our proposed tree-based encoder, we use 64 and 128 for the dimension size of the phrase label embedding. The model parameters are uniformly initialized in $[-0.1, 0.1]$, except that the forget biases are filled with 1.0 as recommended in Józefowicz et al. (2015). Biases, softmax

[1] http://lotus.kuee.kyoto-u.ac.jp/WAT/

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

[3] 16 threads on Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz

Phrase category labels	Description of the label
ADJX	Adjective unsaturated constituent
CX	Complementizer unsaturated constituent
SCX	Subordination conjunction unsaturated constituent
PX	Prepositional unsaturated constituent
PN	Punctuation
DP	Determiner phrase
SCP	Subordination conjunction phrase
S	Sentence
ADVX	Adverb unsaturated constituent
PP	Prepositional phrase
NP	Noun phrase
NX	Noun unsaturated constituent
VP	Verb phrase
ADVP	Adverb phrase
CONJP	Coordination conjunction phrase
COOD	Part of coordination
ADJP	Adjective phrase
CP	Complementizer phrase
VX	Verb unsaturated constituent

Table 5.1: The details of phrase category labels.

weights and BlackOut weights are filled with 0. We shuffle the training data randomly per each epoch. All of the parameters are updated by the plain stochastic gradient descent algorithm with a mini-batch size of 128. The learning rate of stochastic gradient descent is set to 1.0, and we halve it when the perplexity of development data becomes worse. The value of gradient norm clipping [Pascanu et al., 2012] is set to 3.0.

We use a beam search in order to obtain a proper translation sentence with the size of 20 and 5 in the word-based decoder and the character-based decoder, respectively. The maximum length of a generated sentence is set to 100 in the word-based decoder and to 300 in the character-based decoder. Cho et al. (2014a) reported that an RNN-based decoder generates a shorter sentence when using the original beam search. We used the beam search method proposed in Eriguchi et al. (2016) in order to output longer translations. We evaluated the models by the BLEU score [Pa-

	Sentences	Parsed sentences
Training dataset	1,346,946	1,346,946
Development dataset	1,790	1,789
Test dataset	1,812	1,811

Table 5.2: The details of dataset in the ASPEC corpus.

	Vocabulary size
$\|V_{word}\|$ in English	87,796
$\|V_{word}\|$ in Japanese	65,680
$\|V_{character}\|$ in Japanese	3,004

Table 5.3: Vocabulary sizes in the training models.

pineni et al., 2002] and the RIBES score [Isozaki et al., 2010] employed as the official evaluation metrics in WAT'16.

5.4.2 Experimental Results

Table 5.4 shows the experimental results of the character-based models, the word-based models and the baseline SMT systems. BP denotes the brevity penalty in the BLEU score. First, we can see small improvements in the RIBES score of the single tree-to-sequence attention-based NMT models with the character-based decoder using syntactic features, compared to our proposed baseline model. The translations are output by the ensemble of the three models, and we used a simple beam search to confirm how much it effects the BLEU scores in the character-based models. We showed the results of our proposed character-based decoder models by using the beam search method proposed in Eriguchi et al. (2016). We collects the statistics of the relation between the source sentence length (L_s) and the target sentence length (L_t) from training dataset and adds its log probability ($\log p(L_t|L_s)$) as the penalty of the beam score when the model predicts "EOS". The BLEU score is sensitive to the value of BP, and we observe the same trend in that the character-based approaches generate a shorter sentence by the original beam search. As a result, each of the character-based models can generate longer translation by +0.09 BP scores at least than tree-to-sequence attention-based NMT model using the original beam search.

The word-based tree-to-sequece decoder model shows slightly better performance than the

word-based sequence-to-sequence attention-based NMT model [Luong et al., 2015a] in both of the scores. The results of the baseline systems are the ones reported in Nakazawa et al. (2015). Compared to these SMT baselines, each of the character-based models clearly outperforms the phrase-based system in both of the BLEU and RIBES scores. Although the hierarchical phrase-based SMT system and the tree-to-string SMT system outperforms the single character-based model without phrase label inputs by +1.04 and by +1.92 BLEU scores, respectively, our best ensemble of character-based models shows better performance (+5.65 RIBES scores) than the tree-to-string SMT system.

Model	BLEU	Brevity Penalty	RIBES
Character-based decoder			
Our proposed baseline: tree-to-sequence attention-based NMT model	31.52	0.96	79.39
+ phrase label input ($m = 64$)	31.49	0.95	79.51
+ phrase label input ($m = 128$)	31.46	0.95	79.48
Ensemble of the above three models w/ the original beam search	33.21	0.86	81.45
Word-based decoder			
seq-to-sequence attention-based NMT model [Luong et al., 2015a]	34.64	0.93	81.60
tree-to-sequence attention-based NMT model ($d = 512$)	34.91	0.92	81.66
Ensemble of the tree-to-sequence attention-based NMT models [Eriguchi et al., 2016]	36.95	0.92	82.45
Baseline system			
Baseline 1: Phrase-based SMT	29.80	——	69.19
Baseline 2: Hierarchical Phrase-based SMT	32.56	——	74.70
Baseline 3: Tree-to-string SMT	33.44	——	75.80

Table 5.4: The results of our proposed models and the baseline systems.

	Time (msec / sentence)
Word-based decoder	363.7
Character-based decoder	8.8

Table 5.5: Comparison of the times when outputting a sentence.

5.5 Discussion

Table 5.5 shows a comparison of the speeds to predict the next word between the word-based decoder and the character-based decoder when generating a sentence by a beam size of 1. The character-based decoder is about 41 times faster than the word-based decoder. It is because the time to output a word by using a softmax layer is roughly proportional to the vocabulary sizes of the decoders. In addition to the low cost of predicting the outputs in the character-based model, the character-based decoder requires the smaller size of beam search than the word-based model. The word-based decoder requires a beam size of 20 when decoding, but a beam size of 5 is enough for the character-based decoder. It requires smaller beam size for the character-based decoder to find the best hypothesis. We therefore conclude that the character-based model works more efficiently as a translation model than the word-based model in terms of the cost of the outputs.

Some translation examples of our tree-to-sequence attention-based NMT models are shown in Table 5.6. There are two types of source sentences, the ground truth target sentences, and the translated sentences by the word-based model, by the character-based model, and the character-based model using the syntactic features embedded with a dimension size of 64. The words in the same color semantically correspond to each other.

In sentence A, we can see that the character-based models correctly translated the source word "micro" with the characters "マイクロ", while the word-based decoder requires the unknown replacement [Luong et al., 2015b; Jean et al., 2015]. When the word-based model outputs the target word "UNK", the source phrase "micro watt" has the highest attention score ($\alpha = 0.78$) and the source word "micro" has the second highest attention score ($\alpha = 0.16$). The word-based decoder model is successful in outputting the original number ("３８０") in the source side to the target side, and both of the character-based decoder model has also succeeded in predicting a correct sequence of characters "３", "８", and "０" one by one. The training dataset includes the translation of the word "380" into the characters"３８０", so the character-based model can be trained without copy mechanism [Ling et al., 2016; Gu et al., 2016] in this case.

In sentence B, the character-based models successfully translate "low-loss forsterite porcelain"

into “低損失フォルステライト磁器”. The word-based decoder model generates two “UNK”s. The source word “forsterite” (“フォルステライト” in Japanese) has the highest attention score ($\alpha = 0.23$) to the first “UNK”, and the phrase “forsterite porcelain” has the second highest attention score ($\alpha = 0.21$). The second “UNK” is softly alined to the source word “porcelain” (“磁器” in Japanese) with the highest attention score ($\alpha = 0.26$) and to the source phrase “forsterite porcelain” with the second highest attention score ($\alpha = 0.16$).

Source sentence A	The electric power generation was the 380 micro watt.
Ground truth A	発電量は３８０マイクロワットであった。
Word-based	発電は３８０ UNKW であった。
Character-based	発電は３８０マイクロワットであった。
+ label input ($m = 64$)	電力発電は３８０マイクロワットであった。
Source sentence B	This paper describes development outline of low-loss forsterite porcelain.
Ground truth B	低損失フォルステライト磁器の開発概要などをのべた。
Word-based	ここでは，UNK UNK の開発概要を述べた。
Character-based	低損失フォルステライト磁器の開発概要を述べた。
+ label input ($m = 64$)	低損失フォルステライト磁器の開発概要を述べた。

Table 5.6: Translation examples of test data.

5.6 Related Work

There are many NMT architectures: a convolutional network-based encoder [Kalchbrenner and Blunsom, 2013], sequence-to-sequence models [Cho et al., 2014b; Sutskever et al., 2014] and a tree-based encoder [Eriguchi et al., 2016]. The objective of these research efforts focused on how to encode the data in a language into a vector space and to decode the data in another language from the vector space. Sennrich and Haddow (2016) improved the vector space of NMT models by adding linguistic features. The text data is basically considered as the sequence of words.

The word-based NMT models cannot usually cover the whole vocabulary in the corpus. Rare words are mapped into "unknown" words when the NMT models are trained. Luong et al. (2015b) proposed an ex post facto replacement technique for such unknown words, and Jean et al. (2015) replace the unknown word with the source word which has the highest attention score to the unknown word. Sennrich et al. (2016) adopted a sub-word as a unit of the vocabulary for the NMT models and created the sub-word-based vocabulary by the BPE method. The vocabulary based on the sub-words can cover much more words in German and Russian, compared to the vocabulary based on the words. The NMT models trained with the sub-word-based vocabulary performed better than the ones trained on the word-based vocabulary.

Since the smallest units of text data are characters, character-based approaches have been introduced into the fields of NMT. Costa-jussà and Fonollosa (2016) have shown that the character-based encoding by using convolutional networks and the highway network as shown in Kim et al. (2016). Chung et al. (2016) applied the character-based decoding to the NMT models, whose encoder is based on the BPE units. Luong and Manning (2016) have proposed a hybrid NMT model flexibly switching from the word-based to the character-based model. Each character-based NMT model shows better performance than the word-based NMT models. All of theses models are, however, applied to sequence-based NMT models, and there are no results of the character-based decoding applied to tree-based NMT models yet.

5.7 Conclusion

In this chapter, we applied the word-based decoding and the character-based decoding to the tree-to-sequence attention-based neural machine translation model. We also explored the effectiveness by adding the phrase category labels into the tree-based encoder as the previous work focused on the syntactic structures and did not use any syntactic labels in the encoder. The experimental results on English-to-Japanese translation shows that the character-based decoder does not outperform the word-based decoder but exhibits two promising properties: 1) It takes much less time to

compute the softmax layer; and 2) It can translate any word in a sentence.

Chapter 6

Sequence-to-Tree Neural Machine Translation

This chapter has proposed a hybridized model which jointly learn to parse and translate. Our proposed model is trained on the parallel corpus and the parsed tree sentence in the target language, and we let the model to generate only translations and the translations with their parsed tree optionally. This chapter was already published as a conference paper as follows:

Chapter 6 Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. Learning to Parse and Translate in Neural Machine Translation. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 72–78, Vancouver, Canada, August 2017.

6.1 Introduction

Neural Machine Translation (NMT) has enjoyed impressive success without relying on much, if any, prior linguistic knowledge. Some of the most recent studies have for instance demonstrated that NMT systems work comparably to other systems even when the source and target sentences are given simply as flat sequences of characters [Lee et al., 2016; Chung et al., 2016] or statistically, not linguistically, motivated subword units [Sennrich et al., 2016; Wu et al., 2016]. [Shi et al., 2016] recently made an observation that the encoder of NMT captures syntactic properties of a source sentence automatically, indirectly suggesting that explicit linguistic prior may not be necessary.

On the other hand, there have only been a couple of recent studies showing the potential benefit of explicitly encoding the linguistic prior into NMT. [Sennrich and Haddow, 2016] for instance proposed to augment each source word with its corresponding part-of-speech tag, lemmatized form and dependency label. [Eriguchi et al., 2016] instead replaced the sequential encoder with a tree-based encoder which computes the representation of the source sentence following its parse tree. [Stahlberg et al., 2016] let the lattice from a hierarchical phrase-based system guide the decoding process of neural machine translation, which results in two separate models rather than a single end-to-end one. Despite the promising improvements, these explicit approaches are limited in that the trained translation model strictly requires the availability of external tools during inference time. More recently, researchers have proposed methods to incorporate target-side syntax into NMT models. [Alvarez-Melis and Jaakkola, 2017] have proposed a doubly-recurrent neural network that can generate a tree-structured sentence, but its effectiveness in a full scale NMT task is yet to be shown. [Aharoni and Goldberg, 2017] introduced a method to serialize a parsed tree and to train the serialized parsed sentences.

We propose to *implicitly* incorporate linguistic prior based on the idea of multi-task learning [Caruana, 1998; Collobert et al., 2011]. More specifically, we design a hybrid decoder for NMT, called NMT+RNNG[1], that combines a usual conditional language model and a recently proposed recurrent neural network grammars [Dyer et al., 2016]. This is done by plugging in the conventional language model decoder in the place of the buffer in RNNG, while sharing a subset of parameters, such as word vectors, between the language model and RNNG. We train this hybrid model to maximize both the log-probability of a target sentence and the log-probability of a parse action sequence. We use an external parser [Andor et al., 2016] to generate target parse actions, but unlike the previous explicit approaches, we do not need it during test time.

We evaluate the proposed NMT+RNNG on four language pairs ({Jp, Cs, De, Ru}-En). We observe significant improvements in terms of BLEU scores on three out of four language pairs and RIBES scores on all the language pairs.

6.2 Neural Machine Translation

Neural machine translation is a recently proposed framework for building a machine translation system based purely on neural networks. It is often built as an attention-based encoder-decoder network [Cho et al., 2015] with two recurrent networks—encoder and decoder—and an attention model. The encoder, which is often implemented as a bidirectional recurrent network with long short-term memory units [Hochreiter and Schmidhuber, 1997] or gated recurrent units [Cho et al.,

[1]Our code is available at `https://github.com/tempra28/nmtrnng`.

2014b], first reads a source sentence represented as a sequence of words $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$. The encoder returns a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_N)$. Each hidden state $\boldsymbol{h}_i \in \mathbb{R}^{2d_V \times 1}$ is a concatenation of those from the forward and backward recurrent network: $\boldsymbol{h}_i = \left[\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i\right]$, where

$$\overrightarrow{\boldsymbol{h}}_i = \overrightarrow{f}_{\text{enc}}(\overrightarrow{\boldsymbol{h}}_{i-1}, V_x(x_i)), \tag{6.1}$$

$$\overleftarrow{\boldsymbol{h}}_i = \overleftarrow{f}_{\text{enc}}(\overleftarrow{\boldsymbol{h}}_{i+1}, V_x(x_i)). \tag{6.2}$$

$V_x(x_i) \in \mathbb{R}^{d_V \times 1}$ refers to the word vector of the i-th source word. $\overrightarrow{f}_{\text{enc}}$ and $\overrightarrow{f}_{\text{enc}}$ denote a recurrent activation function, such as LSTM or GRU, respectively.

The decoder is implemented as a conditional recurrent language model which models the target sentence, or translation, as

$$\log p(\boldsymbol{y}|\boldsymbol{x}) = \sum_j \log p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}), \tag{6.3}$$

where $\boldsymbol{y} = (y_1, \ldots, y_M)$. Each of the conditional probabilities in the r.h.s is computed by

$$p(y_j = y|\boldsymbol{y}_{<j}, \boldsymbol{x}) = softmax(\boldsymbol{W}_y^\top \tilde{\boldsymbol{s}}_j), \tag{6.4}$$

$$\tilde{\boldsymbol{s}}_j = \tanh(\boldsymbol{W}_c[\boldsymbol{s}_j; \boldsymbol{c}_j]), \tag{6.5}$$

$$\boldsymbol{s}_j = f_{\text{dec}}(\boldsymbol{s}_{j-1}, [V_y(y_{j-1}); \tilde{\boldsymbol{s}}_{j-1}]), \tag{6.6}$$

where f_{dec} is a recurrent activation function, such as LSTM or GRU, and $\boldsymbol{W}_y$ is the output word vector of the word y.

$\boldsymbol{c}_j$ is a time-dependent context vector that is computed by the attention model using the sequence $\boldsymbol{h}$ of hidden states from the encoder. The attention model first compares the current hidden state $\boldsymbol{s}_j$ against each of the hidden states and assigns a scalar score [Luong et al., 2015a] calculated as follows:

$$\beta_{i,j} = \exp(\boldsymbol{h}_i^\top \boldsymbol{W}_d \boldsymbol{s}_j), \tag{6.7}$$

where $\boldsymbol{W}_d$ is a conversion matrix. These scores are then normalized across the hidden states to sum to 1, that is:

$$\alpha_{i,j} = \frac{\beta_{i,j}}{\sum_i \beta_{i,j}}. \tag{6.8}$$

The time-dependent context vector is then a weighted-sum of the hidden states with these attention weights computed as:

$$\boldsymbol{c}_j = \sum_i \alpha_{i,j} \boldsymbol{h}_i. \tag{6.9}$$

6.3 Recurrent Neural Network Grammars

6.3.1 Stack LSTM

The stack LSTM units [Dyer et al., 2015] have been proposed to compute the tree-structured LSTM network by using the stack structures. Now we consider recurrent neural networks applied with LSTM units. The current hidden unit $\boldsymbol{h}_t$ is computed by using the previous hidden unit $\boldsymbol{h}_{t-1}$ and the current unit $\boldsymbol{h}_t$ as follows:

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{v}_t), \tag{6.10}$$

where $\boldsymbol{h}_t, \boldsymbol{h}_{t-1} \in \mathbb{R}^{d_H \times 1}$, and $\boldsymbol{v}_t \in \mathbb{R}^{d_V \times 1}$ are the d_H-dimensional vectors and a d_V-dimensional input vector, respectively. f is an LSTM function.

The stack LSTM unit is a type of the LSTM units. Unlike the LSTM units in recurrent neural networks described above, the stack LSTM units are stored in the stack structure and computed by using the top of the unit in the stack ($\boldsymbol{h}_{TOP} \in \mathbb{R}^{d_H \times 1}$) as follows:

$$\boldsymbol{h}_t = g(\boldsymbol{h}_{TOP}, \boldsymbol{v}_t), \tag{6.11}$$

where $\boldsymbol{h}_t, \boldsymbol{h}_{TOP} \in \mathbb{R}^{d_H \times 1}$, and $\boldsymbol{v}_t \in \mathbb{R}^{d_V \times 1}$ are the d_H-dimensional vectors and a d_V-dimensional input vector, respectively. g is a stack LSTM function, which is equal to the LSTM function. There are two kinds of the operations of "*POP*" and "*PUSH*" in the stack LSTM units.

POP

Move the *TOP* to the previous stack LSTM unit.

PUSH

Compute the next stack LSTM unit by using the the stack LSTM unit specified with the *TOP*.

Figure 6.1 shows how the two types of the operations work in the stack LSTM units. We illustrate the top of the units in the stack by making the unit with an arrow named with "*TOP*" in Figure 6.1.

6.3.2 Recurrent Neural Network Grammars

A recurrent neural network grammar [Dyer et al., 2016] is a probabilistic syntax-based language model. Unlike a usual recurrent language model [Mikolov et al., 2010], an RNNG simultaneously models both tokens and their tree-based composition. This is done by having a (output) buffer, stack and action history, each of which is implemented as a stack LSTM (sLSTM) [Dyer et al.,

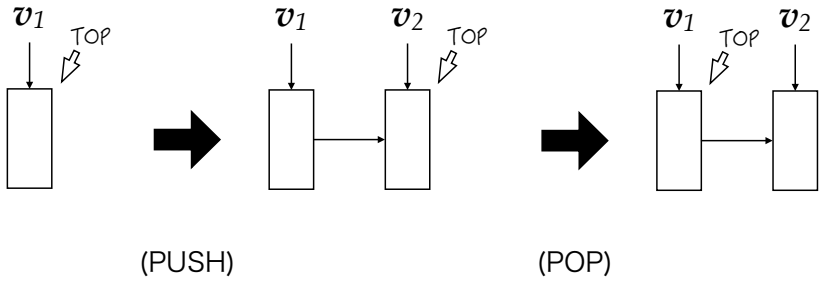


Figure 6.1: Two types of the operations in the stack LSTM units.

2015]. At each time step, the action stack LSTM predicts the next action based on the (current) hidden states of the buffer, stack and action sLSTM. That is,

$$p(a_t = a|\boldsymbol{a}_{<t}) \propto \exp^{\boldsymbol{W}_a^\top f_{\text{action}}(\boldsymbol{h}_t^{\text{buffer}}, \boldsymbol{h}_t^{\text{stack}}, \boldsymbol{h}_t^{\text{action}})}, \tag{6.12}$$

where $\boldsymbol{W}_a$ is the vector of the action a. If the selected action is *shift*, the word at the beginning of the buffer is moved to the stack. When the *reduce* action is selected, the top-two words in the stack are reduced to build a partial tree. Figure 6.2 shows the building process of the syntactic vectors. Each action corresponds to the predefined sequence of two operations of *POP* and *PUSH*. Here, $\boldsymbol{p}(whitedog)$ denotes a dependency phrase vectors of "white dog". Additionally, the action may be one of many possible non-terminal symbols, in which case the predicted non-terminal symbol is pushed to the stack.

The hidden states of the buffer, stack and action sLSTM are correspondingly updated by

$$\begin{aligned}
\boldsymbol{h}_t^{\text{buffer}} &= \text{StackLSTM}(\boldsymbol{h}_{\text{top}}^{\text{buffer}}, V_y(y_{t-1})), \\
\boldsymbol{h}_t^{\text{stack}} &= \text{StackLSTM}(\boldsymbol{h}_{\text{top}}^{\text{stack}}, \boldsymbol{r}_t), \\
\boldsymbol{h}_t^{\text{action}} &= \text{StackLSTM}(\boldsymbol{h}_{\text{top}}^{\text{action}}, V_a(a_{t-1})),
\end{aligned} \tag{6.13}$$

where V_y and V_a are functions returning the target word and action vectors. The input vector $\boldsymbol{r}_t$ of the sLSTM is computed recursively by

$$\boldsymbol{r}_t = \tanh(\boldsymbol{W}_r[\boldsymbol{r}^d; \boldsymbol{r}^p; V_a(a_t)]),$$

where $\boldsymbol{r}^d$ and $\boldsymbol{r}^p$ are the corresponding vectors of the parent and dependent phrases, respectively [Dyer et al., 2015]. This process is iterated until a complete parse tree is built. Note that the original paper of RNNG [Dyer et al., 2016] uses constituency trees, but we employ dependency

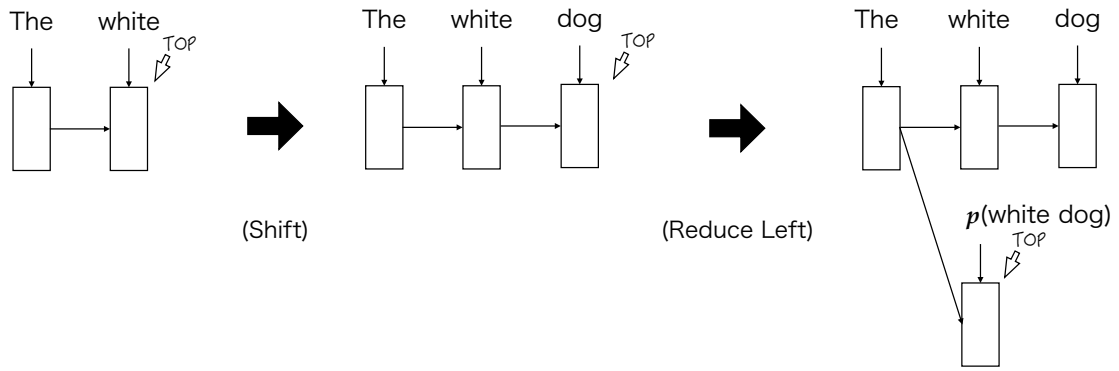


Figure 6.2: How to compute the syntactic structured sentence vectors by using Stack LSTM units in a dependency parsed sentence.

trees in this study. Figure 6.3 is an example of a universal dependency tree. Both types of trees are represented as a sequence of the three types of actions in a transition-based parsing model.

When the complete sentence is provided, the buffer simply summarizes the shifted words. When the RNNG is used as a generator, the buffer further generates the next word when the selected action is shift. The latter can be done by replacing the buffer with a recurrent language model, which is the idea on which our proposal is based.

6.4 Learning to Parse and Translate

6.4.1 NMT+RNNG

Our main proposal in this study is to hybridize the decoder of the neural machine translation and the RNNG. We continue from the earlier observation that we can replace the buffer of RNNG to a recurrent language model that simultaneously summarizes the shifted words as well as generates future words. We replace the RNNG's buffer with the neural translation model's decoder in two steps. Figure 6.4 shows the outline of our proposed model.

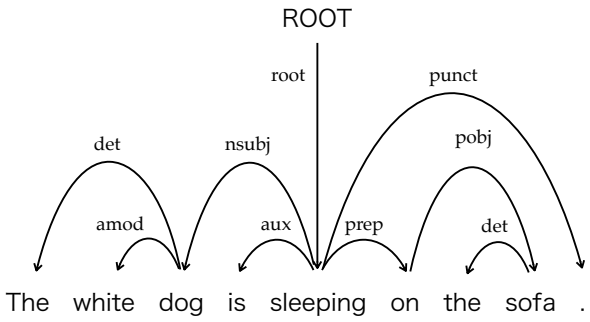


Figure 6.3: An example of universal dependency tree structure of a sentence “The white dog is sleeping on the sofa.”.

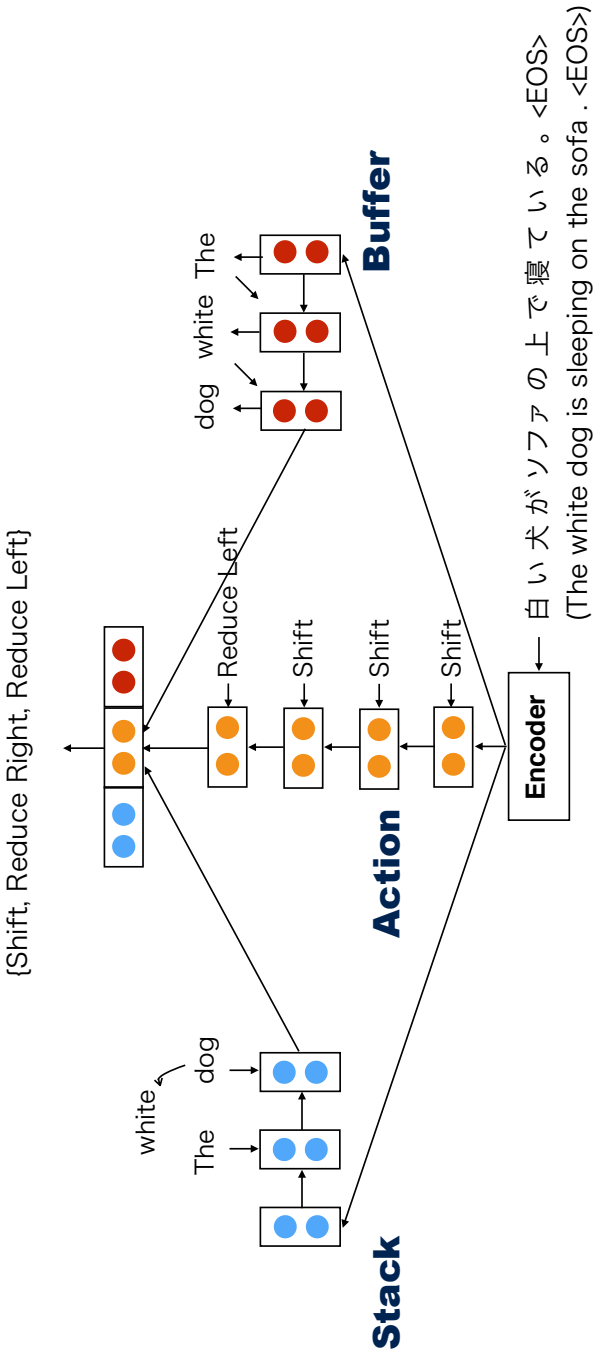


Figure 6.4: Proposed model: Sequence-to-tree neural machine translation model.

Construction First, we replace the hidden state of the buffer $\boldsymbol{h}^{\text{buffer}}$ (in Eq. (6.13)) with the hidden state of the decoder of the attention-based neural machine translation from Eq. (6.6). As is clear from those two equations, both the buffer sLSTM and the translation decoder take as input the previous hidden state ($\boldsymbol{h}^{\text{buffer}}_{\text{top}}$ and $\boldsymbol{s}_{j-1}$, respectively) and the previously decoded word (or the previously shifted word in the case of the RNNG's buffer), and returns its summary state. The only difference is that the translation decoder additionally considers the state $\tilde{\boldsymbol{s}}_{j-1}$. Once the buffer of the RNNG is replaced with the NMT decoder in our proposed model, the NMT decoder is also under control of the actions provided by the RNNG.[2] Second, we let the next word prediction of the translation decoder as a generator of RNNG. In other words, the generator of RNNG will output a word, when asked by the shift action, according to the conditional distribution defined by the translation decoder in Eq. (6.4). Once the buffer sLSTM is replaced with the neural translation decoder, the action sLSTM naturally takes as input the translation decoder's hidden state when computing the action conditional distribution in Eq. (6.12). We call this hybrid model *NMT+RNNG*.

Learning and Inference After this integration, our hybrid NMT+RNNG models the conditional distribution over all possible pairs of translation and its parse given a source sentence, i.e., $p(\boldsymbol{y}, \boldsymbol{a}|\boldsymbol{x})$. Assuming the availability of parse annotation in the target-side of a parallel corpus, we train the whole model jointly to maximize $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{a})\sim\text{data}}\left[\log p(\boldsymbol{y}, \boldsymbol{a}|\boldsymbol{x})\right]$. In doing so, we notice that there are two separate paths through which the neural translation decoder receives error signal. First, the decoder is updated in order to maximize the conditional probability of the correct next word, which has already existed in the original neural machine translation. Second, the decoder is updated also to maximize the conditional probability of the correct parsing action, which is a novel learning signal introduced by the proposed hybridization. Furthermore, the second learning signal affects the encoder as well, encouraging the whole neural translation model to be aware of the syntactic structure of the target language. Later in the experiments, we show that this additional learning signal is useful for translation, even though we discard the RNNG (the stack and action sLSTM units) in the inference time.

6.4.2 Knowledge Distillation for Parsing

A major challenge in training the proposed hybrid model is that there is not a parallel corpus augmented with gold-standard target-side parse, and vice versa. In other words, we must either

[2]The j-th hidden state in Eq. (6.6) is calculated only when the action (*shift*) is predicted by the RNNG. This is why our proposed model can handle the sequences of words and actions which have different lengths.

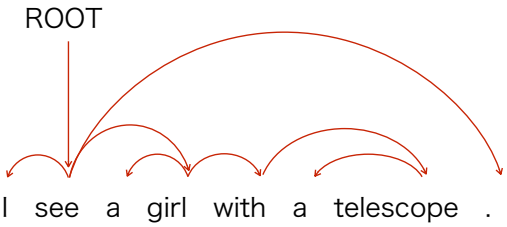
	Traininig	Development	Test	Voccabulary (*src*, *tgt, act*)
Cs-En	134,453	2,656	2,999	(33,867, 27,347, 82)
De-En	166,313	2,169	2,999	(33,820, 30,684, 80)
Ru-En	131,492	2,818	2,998	(32,442, 27,979, 82)
Jp-En	100,000	1,790	1,812	(23,509, 28,591, 80)

Table 6.1: Statistics of parallel corpora.

parse the target-side sentences of an existing parallel corpus or translate sentences with existing gold-standard parses. As the target task of the proposed model is translation, we start with a parallel corpus and annotate the target-side sentences. It is however costly to manually annotate any corpus of reasonable size [Alonso et al., 2016].

We instead resort to noisy, but automated annotation using an existing parser. This approach of automated annotation can be considered along the line of recently proposed techniques of knowledge distillation [Hinton et al., 2015] and distant supervision [Mintz et al., 2009]. In knowledge distillation, a teacher network is trained purely on a training set with ground-truth annotations, and the annotations predicted by this teacher are used to train a student network, which is similar to our approach where the external parser could be thought of as a teacher and the proposed hybrid network's RNNG as a student. On the other hand, what we propose here is a special case of distant supervision in that the external parser provides noisy annotations to otherwise an unlabeled training set.

Specifically, we use SyntaxNet, released by [Andor et al., 2016], on a target sentence.[3] We convert a parse tree into a sequence of one of three transition actions (SHIFT, REDUCE-L, REDUCE-R). We label each REDUCE action with a corresponding dependency label and treat it as a more fine-grained action. Figure 6.5 shows an example of the conversion of a dependency parsed tree to a sequence of three types of actions. The words within the brackets represent terminal words.



Shift (I), Shift (see), Reduce Left, Shift (a), Shift (girl), Reduce Left,
Shift (with), Shift (a), Shift (telescope), Reduce Left, Reduce Right,
Reduce Right, Reduce Right, Shift (.), Reduce Left, Shift (ROOT), Reduce Left

Figure 6.5: An example of converting a dependency parsed tree to a sequence of actions.

6.5 Experiments

6.5.1 Language Pairs and Corpora

We compare the proposed NMT+RNNG against the baseline model on four different language pairs–Jp-En, Cs-En, De-En and Ru-En. The basic statistics of the training data are presented in Table 6.1. We mapped all the low-frequency words to the unique symbol "UNK" and inserted a special symbol "EOS" at the end of both source and target sentences.

Jp

We use the ASPEC corpus ("train1.txt") from the WAT'16 Jp-En translation task. We tokenize each Japanese sentence with *KyTea* [Neubig et al., 2011] and preprocess according to the recommendations from WAT'16 [WAT, 2016]. We use the first 100K sentence pairs of length shorter than 50 for training. The vocabulary is constructed with all the unique tokens that appear at least twice in the training corpus. We use "dev.txt" and "test.txt" provided by WAT'16 respectively as development and test sets.

[3]When the target sentence is parsed as data preprocessing, we use all the vocabularies in a corpus and do not cut off any words. We use the plain SyntaxNet and do not train it furthermore.

Cs, De and Ru

We use News Commentary v8. We removed noisy metacharacters and used the tokenizer from Moses [Koehn et al., 2007] to build a vocabulary of each language using unique tokens that appear at least 6, 6 and 5 times respectively for Cs, Ru and De. The target-side (English) vocabulary was constructed with all the unique tokens appearing more than three times in each corpus. We also excluded the sentence pairs which include empty lines in either a source sentence or a target sentence. We only use sentence pairs of length 50 or less for training. We use "newstest2015" and "newstest2016" as development and test sets respectively.

6.5.2 Models, Learning and Inference

In all our experiments, each recurrent network has a single layer of LSTM units of 256 dimensions, and the word vectors and the action vectors are of 256 and 128 dimensions, respectively. To reduce computational overhead, we use BlackOut [Ji et al., 2016] with 2000 negative samples and $\alpha = 0.4$. When employing BlackOut, we shared the negative samples of each target word in a sentence in training time [Hashimoto and Tsuruoka, 2017], which is similar to the previous work [Zoph et al., 2016]. For the proposed NMT+RNNG, we share the target word vectors between the decoder (buffer) and the stack sLSTM.

Each weight is initialized from the uniform distribution $[-0.1, 0.1]$. The bias vectors and the weights of the softmax and BlackOut are initialized to be zero. The forget gate biases of LSTM units and Stack-LSTM units are initialized to 1 as recommended in [Józefowicz et al., 2015]. We use stochastic gradient descent with minibatches of 128 examples. The learning rate starts from 1.0, and is halved each time the perplexity on the development set increases. We clip the norm of the gradient [Pascanu et al., 2012] with the threshold set to 3.0 (2.0 for the baseline models on Ru-En and Cs-En to avoid NaN and Inf). When the perplexity of development data increased in training time, we halved the learning rate of stochastic gradient descent and reloaded the previous model. The RNNG's stack computes the vector of a dependency parse tree which consists of the generated target words by the buffer. Since the complete parse tree has a "ROOT" node, the special token of the end of a sentence ("EOS") is considered as the ROOT. We use beam search in the inference time, with the beam width selected based on the development set performance.

It took about 15 minutes per epoch and about 20 minutes respectively for the baseline and the proposed model to train a full JP-EN parallel corpus in our implementation.[4]

[4]We run all the experiments on multi-core CPUs (10 threads on Intel(R) Xeon(R) CPU E5-2680 v2 @2.80GHz)

	De-En	Ru-En	Cs-En	Jp-En
BLEU				
NMT	16.61	12.03	11.22	17.88
NMT+RNNG	16.41	**12.46**†	**12.06**†	**18.84**†

Table 6.2: BLEU scores by the baseline and proposed models on the test set.

	De-En	Ru-En	Cs-En	Jp-En
RIBES				
NMT	73.75	69.56	69.59	71.27
NMT+RNNG	**75.03**†	**71.04**†	**70.39**†	**72.25**†

Table 6.3: RIBES scores by the baseline and proposed models on the test set.

6.5.3 Results and Analysis

We use the bootstrap resampling method from [Koehn, 2004] to compute the statistical significance. We use † to mark those significant cases with $p < 0.005$. In Table 6.5.3 and Table 6.5.3, we report the translation qualities of the tested models on all the four language pairs. We report both BLEU [Papineni et al., 2002] and RIBES [Isozaki et al., 2010]. Except for De-En, measured in BLEU, we observe the statistically significant improvement by the proposed NMT+RNNG over the baseline model. It is worthwhile to note that these significant improvements have been achieved *without* any additional parameters nor computational overhead in the inference time.

Ablation Since each component in RNNG may be omitted, we ablate each component in the proposed NMT+RNNG to verify their necessity.[5] As shown in Table 6.4, we see that the best performance could only be achieved when all the three components were present. Removing the stack had the most adverse effect, which was found to be the case for parsing as well by [Kuncoro et al., 2017].

[5]Since the buffer is the decoder, it is not possible to completely remove it. Instead we simply remove the dependency of the action distribution on it.

Jp-En (Dev)	BLEU
NMT+RNNG	18.60
w/o Buffer	18.02
w/o Action	17.94
w/o Stack	17.58
NMT	17.75

Table 6.4: Effect of each component in RNNG.

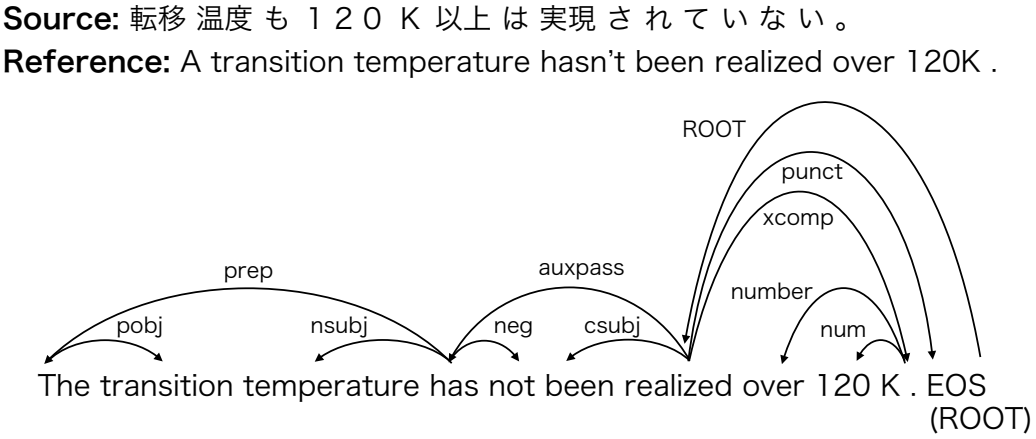


Figure 6.6: An example of translation and its dependency relations obtained by our proposed model.

Generated Sentences with Parsed Actions The decoder part of our proposed model consists of two components: the NMT decoder to generate a translated sentence and the RNNG decoder to predict its parsing actions. The proposed model can therefore output a dependency structure along with a translated sentence. Figure 6.6, 6.7 and 6.8 show the examples of Jp-En translation in the development dataset and its dependency parse tree obtained by the proposed model. The special symbol ("EOS") is treated as the root node ("ROOT") of the parsed tree. The translated sentence was generated by using beam search, which is the same setting of NMT+RNNG shown in Table 6.4. The parsing actions were obtained by greedy search. The resulting dependency structure is mostly correct but contains a few errors; for example, dependency relation between "The" and " transition" should not be "pobj" in Figure 6.6.

Figure 6.7: An example of translation and its dependency relations obtained by our proposed model.

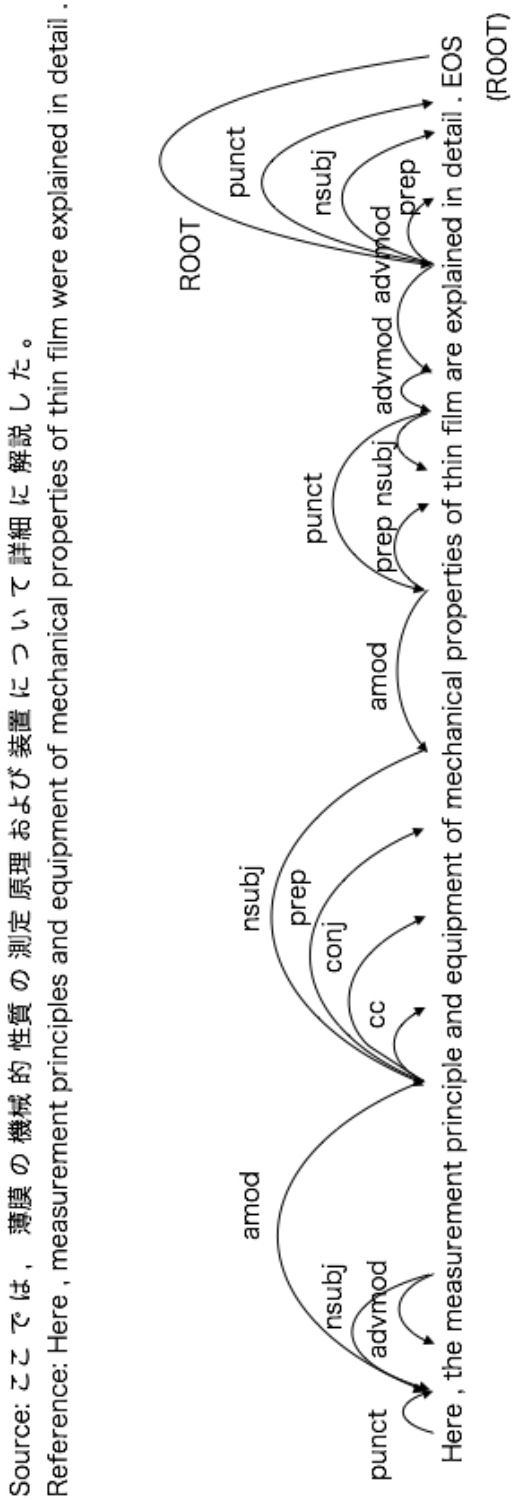


Figure 6.8: An example of translation and its dependency relations obtained by our proposed model.

6.6 Conclusion

We propose a hybrid model, to which we refer as NMT+RNNG, that combines the decoder of an attention-based neural translation model with the RNNG. This model learns to parse and translate simultaneously, and training it encourages both the encoder and decoder to better incorporate linguistic priors. Our experiments confirmed its effectiveness on four language pairs ({Jp, Cs, De, Ru}-En). The RNNG can in principle be trained without ground-truth parses, and this would eliminate the need of external parsers completely. We leave the investigation into this possibility for future research.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we focus on the syntactic structure inherited in a language. We have proposed two types of the syntax-based neural machine translation models which incorporate syntactic structures into the sequence-to-sequence neural machine translation models.

We focus on phrase structure in a source language. Phrase structures build a phrase component in a bottom-up fashion. We have proposed a tree-based encoder to compute phrase vectors from the existing sequential encoder. Following the binary tree of the phrase structures obtained by the external parser tools, we build a tree-based encoder. Experimental results have shown that our proposed tree-to-sequence NMT model improves translation accuracy better than the existing sequence-to-sequence NMT model. We also confirmed the proposed model succeeded in flexibly aligning a target word with a source word or a source phrase in the English-to-Japanese translation task, where it is well known that the syntactic information improves the machine translation models.

The character-based decoding is well known to easily solve the vocabulary coverage problem in the NMT models because any types of words can be represented as a sequence of words. We applied the character-based decoding approach in the tree-to-sequence NMT model. We found that the character-based decoding approach require the NMT model generate a longer sentence because the NMT decoder favors a shorter sentence at test time, while the character-based decoding solves the vocabulary coverage problem the word-based decoding approach has.

We also focus on universal dependency tree structure to capture the dependency relations between the words in a sentence. The dependency relations capture the different information from phrase structures. We have proposed the multi-task model of the NMT model and recurrent

neural network grammars model which jointly learn to parse and translate. Experimental results on four language translation tasks have shown that the proposed method improved the translation accuracy in every language pairs. We also confirmed that our proposed model optionally generates a translation and a translation with its parsed tree.

7.2 Future of Neural Machine Translation

This thesis has focused on the syntactic structures inherited in languages and developed the syntax-based NMT models. We itemize the future work of neural machine translation.

To build a document-level translation model

The current machine translation models have been developed as a sentence-level translation systems. The learning algorithm assumes a training corpus which consists of sentence-level translation pairs, hence the current machine translation model have no mechanism to learn the meaning of a word from the context beyond the sentence. The contextual information has been studied in the area of discourse parsing [Li et al., 2014]. We have already shown the NMT models which uses a syntactic structured data and will extend the approach into the document-level training corpus.

To utilize the contextual information

The NMT models are mostly trained on a parallel corpus as a training data set, and the model parameters are learned from the text corpus. There are many studies which succeeded in generating a sentence from an image in a image caption task [Xu et al., 2015]. The recent multi-task model which composed of eight tasks suggests that the image generation task improves the parsing task in spite of no direct relations observed between two tasks [Kaiser et al., 2017]. The image data can be useful for building a common semantic space without multi-lingual parallel corpus [Nakayama and Nishida, 2017]. It suggests that even when increasing the textual training data, the model still lacks the contextual information as the world knowledge. The different types of the data such as image and sounds indirectly improve the model performance and multi-task learning is useful for the purpose.

To transfer the learned multilingual representations to other NLP tasks

Johnson et al. (2017) have reported that multi-lingual NMT model learns the interlingual information from the training corpus. There are a few studies [Hill et al., 2015; Schwenk et al., 2017] which reported the different trends of the word embedding vectors learned in a single language-based model such as word2vec [Mikolov et al., 2013a] and Glove

vectors [Pennington et al., 2014] and the conditional language-based model such as the NMT models. McCann et al. (2017) employed the representations learned from the NMT model as the input features in the other natural language tasks and succeeded in improving the model performance in several text classification tasks. These previous studies transfer the English representations to the tasks in English, and it is not clear that we apply the same method in the multilingual set up. There is also still an open question to the reason why these multilingual representation or the representations learned in natural language inference task [Conneau et al., 2017] are useful for the other NLP tasks.

References

[Aharoni and Goldberg2017] Roee Aharoni 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.

[Alonso et al.2016] Héctor Martínez Alonso, Djamé Seddah, and Benoît Sagot. 2016. From noisy questions to minecraft texts: Annotation challenges in extreme syntax scenario. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 13–23, December.

[Alvarez-Melis and Jaakkola2017] David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *Proceedings of International Conference on Learning Representations 2017*.

[Andor et al.2016] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2442–2452.

[Bahdanau et al.2015] 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*.

[Bengio et al.2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

[Brown et al.1993] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguist.*, 19(2):263–311, June.

[Caruana1998] Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

[Chen and Manning2014] Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

[Cheng et al.2016] 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*.

[Chiang2007] David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228, June.

[Cho et al.2014a] 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)*.

[Cho et al.2014b] 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.

[Cho et al.2015] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886.

[Chung et al.2016] Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1693–1703.

[Cohn et al.2016] 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*.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

[Conneau et al.2017] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 681–691.

[Costa-jussà and Fonollosa2016] R. Marta Costa-jussà and R. José A. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 357–361.

[Dai et al.2014] Andrew Dai, Christopher Olah, Quoc Le, and Greg Corrado. 2014. Document Embedding with Paragraph Vectors. In *Proceedings of Deep Learning Workshop at the 2014 Conference on Neural Information Processing Systems*.

[Denkowski and Lavie2014] 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*.

[Dyer et al.2015] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and A. Noah Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 334–343.

[Dyer et al.2016] Chris Dyer, 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.

[Elman1990] Jeffrey L. Elman. 1990. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211.

[Eriguchi et al.2016] Akiko Eriguchi, 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.

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

[Goodfellow et al.2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 2672–2680.

[Gu et al.2016] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

[Gutmann and Hyvärinen2012] 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.

[Hannun et al.2014] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. 2014. Deep speech: Scaling up end-to-end speech recognition. *arxiv preprint arXiv: 1412.5567*.

[Hashimoto and Tsuruoka2017] Kazuma Hashimoto 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.

[Hill et al.2015] Felix Hill, Kyunghyun Cho, Sébastien Jean, Coline Devin, and Yoshua Bengio. 2015. Embedding word similarity with neural machine translation. *Proceedings of the workshop at International Conference on Learning Representations 2015*.

[Hinton et al.2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

[Isozaki et al.2010] 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.

[Jean et al.2015] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for 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 1–10.

[Ji et al.2016] 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*.

[Johnson et al.2017] Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Vigas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

[Józefowicz et al.2015] 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.

[Kaiser et al.2017] Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. arXiv preprint arxiv:1706.05137.

[Kalchbrenner and Blunsom2013] 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.

[Kim et al.2016] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *The Thirtieth AAAI Conference on Artificial Intelligence, AAAI 2016*.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

[Koehn2004] Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

[Krizhevsky et al.2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Kuncoro et al.2017] Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258.

[Lee et al.2015] 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.

[Lee et al.2016] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*.

[Li et al.2014] Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069.

[Li et al.2015] Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314, Lisbon, Portugal, September. Association for Computational Linguistics.

[Ling et al.2016] Wang Ling, Phil Blunsom, Edward Grefenstette, Moritz Karl Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code gen-

eration. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609.

[Liu et al.2006] 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.

[Luong and Manning2016] Minh-Thang Luong and D. Christopher Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1054–1063.

[Luong et al.2015a] 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.

[Luong et al.2015b] 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.

[McCann et al.2017] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6297–6308.

[Mikolov et al.2010] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pages 1045–1048.

[Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at the International Conference on Learning Representations*.

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

[Mikolov et al.2013c] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic Regularities in Continuous Space Word Representations, booktitle = Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pages 746–751.

[Mintz et al.2009] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

[Miyao and Tsujii2008] Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature Forest Models for Probabilistic HPSG Parsing. *Computational Linguistics*, 34(1):35–80.

[Nagao1984] Makoto Nagao. 1984. A framework of a mechanical translation between japanese and english by analogy principle. In *Proc. Of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180.

[Nakayama and Nishida2017] Hideki Nakayama and Noriki Nishida. 2017. Zero-resource machine translation by multimodal encoder-decoder network with multimedia pivot. In *Journal of Machine Translation*, volume 31, pages 49–64.

[Nakazawa et al.2015] 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.

[Nakazawa et al.2016a] Toshiaki Nakazawa, Hideya Mino, Chenchen Ding, Isao Goto, Graham Neubig, Sadao Kurohashi, and Eiichiro Sumita. 2016a. Overview of the 3rd workshop on asian translation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*.

[Nakazawa et al.2016b] Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016b. Aspec: Asian scientific paper excerpt corpus. In *Proceedings of the 10th Conference on International Language Resources and Evaluation (LREC2016)*.

[Neubig and Duh2014] 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.

[Neubig et al.2011] 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.

[Neubig et al.2015] 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.

[Papineni et al.2002] 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.

[Pascanu et al.2012] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *arxiv preprint arXiv: 1211.5063*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Pouget-Abadie et al.2014] 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.

[Sag et al.2003] 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.

[Schwenk et al.2017] Holger Schwenk, Ke Tran, Orhan Firat, and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 157–167.

[Sennrich and Haddow2016] Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, August.

[Sennrich et al.2016] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

[Shi et al.2016] Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534.

[Shrivastava et al.2017] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. 2017. Learning from simulated and unsupervised images through adversarial training. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2242–2251.

[Snover et al.2006] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

[Socher et al.2010] Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *NIPS Workhop on Deep Learning and Unsupervised Feature Learning Workshop 2010*.

[Socher et al.2011] Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24*, pages 801–809.

[Socher et al.2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

[Stahlberg et al.2016] Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 299–305.

[Sutskever et al.2014] 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.

[Tai et al.2015] 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.

[van den Oord et al.2016] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. In *arXiv preprint arXiv:1609.03499*.

[WAT2016] WAT. 2016. http://lotus.kuee.kyoto-u.ac.jp/WAT/baseline/dataPreparationJE.html.

[Wu et al.2016] Yonghui Wu, 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*.

[Xu et al.2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057.

[Yamada and Knight2001] 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.

[Zhu2015] Zhongyuan Zhu. 2015. Evaluating Neural Machine Translation in English-Japanese Task. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 61–68.

[Zoph and Knight2016] 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*.

[Zoph et al.2016] Barret Zoph, Ashish Vaswani, Jonathan May, and Kevin Knight. 2016. Simple, Fast Noise-Contrastive Estimation for Large RNN Vocabularies. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1217–1222.

Publication list

Reviewed International Conference

1. Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. Tree-to-Sequence Attentional Neural Machine Translation. *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 823–833, Berlin, Germany, August 2016.

2. Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. Learning to Parse and Translate in Neural Machine Translation. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 72–78, Vancouver, Canada, August 2017.

Reviewed International Workshop

1. Yuchen Qiao, Kazuma Hashimoto, Akiko Eriguchi, Haxia Wang, Dongsheng Wang, Yoshimasa Tsuruoka, and Kenjiro Taura. Cache Friendly Parallelization of Neural Encoder-Decoder Models without Padding on Multi-core Architecture. *In Proceedings of 2017 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2017*, pp. 437–440, Orlando / Buena Vista, USA, June 2017.

Non-reviewed International Workshop

1. Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. Character-based Decoding in Tree-to-Sequence Attention-based Neural Machine Translation. *In Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pp. 175–183, Osaka, Japan, December 2016.

2. Kazuma Hashimoto, Akiko Eriguchi, and Yoshimasa Tsuruoka. Domain Adaptation and Attention-Based Unknown Word Replacement in Chinese-to-Japanese Neural Machine Translation. *In Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pp. 75–83, Osaka, Japan, December 2016.

Non-reviewed national conference

1. 江里口瑛子, 橋本和真, 鶴岡慶雅. 句構造へのアテンションに基づくニューラル機械翻訳モデル. 言語処理学会 第 22 回年次大会発表論文集, pp. 697–700, 仙台, 2016 年 3 月. (若手奨励賞)

2. 江里口瑛子, Kyunghyun Cho, 鶴岡慶雅. 目的言語における係り受け構造を考慮したニューラル機械翻訳. 言語処理学会 第 23 回年次大会発表論文集, つくば, 2017 年 3 月.

Acknowledgement

First of all, I wish to express my deep gratitude to my supervisor Professor Yoshimasa Tsuruoka. He always makes the time to discuss my research. The fruitful discussions and his helpful comments let me challenge what had never been done before. I also deeply grateful to Professor Issei Sato, who introduced me Tsuruoka Lab. Without his cordial suggestion to visit Tsuruoka Laboratory, I would not have entered phD course.

I was often motivated by my laboratory mates' doing their unique researches; Kazuma Hashimoto, Hirotaka Kameko, and Naoki Mizukami. Especially thanks to Kazuma's great suggestions, I had a chance to implement the neural machine translation models almost from scratch in C++.

It was a great opportunity for me to visit Professor Kyungyhun Cho at New York University, and I stayed in an intellectually stimulating environment. I also could not forget the people and the things I encountered in New York, which gave me lots of fresh perspectives on my life.

The studies in the thesis were supported by Japan Society for Promotion of Science (JSPS) KAKENHI Grant Number 15J12597. I thank a lot to the stuff at JSPS for their devoted supports.

I also showed my great appreciation to my best friends: Miwako, Yurie, and Hikari. As their open-mindedness bonds us closely at any time and they always cheer me up, I could not have finished my phD course without them.

Finally, I would like to show all of my heartfelt thanks to my family. They are indispensable for my life. My parents and brother have supported and helped me in a variety of manners. As they always trust me to accomplish what I decided to do, I have done lots of things so far and I will try to do the next things after my phD course.