Chunk-based Decoder for Neural Machine Translation

Shonosuke Ishiwatar[†∗] **Jingtao Yao**[‡∗] **Shujie Liu**[§] **Mu Li**[§] **Ming Zhou**[§]
Naoki Yoshinaga[¶] **Masaru Kitsuregawa**[‖¶] **Weijia Jia**[‡]
† The University of Tokyo ‡ Shanghai Jiao Tong University § Microsoft Research Asia
¶ Institute of Industrial Science, the University of Tokyo ‖ National Institute of Informatics
[†¶‖]{ishiwatari, ynaga, kitsure}@tkl.iis.u-tokyo.ac.jp
[‡]{yjt1995@, jia-wj@cs.}sjtu.edu.cn
[§]{shujliu, muli, mingzhou}@microsoft.com

Abstract

Chunks (or phrases) once played a pivotal role in machine translation. By using a chunk rather than a word as the basic translation unit, local (intra-chunk) and global (inter-chunk) word orders and dependencies can be easily modeled. The chunk structure, despite its importance, has not been considered in the decoders used for neural machine translation (NMT). In this paper, we propose chunk-based decoders for NMT, each of which consists of a chunk-level decoder and a word-level decoder. The chunk-level decoder models global dependencies while the word-level decoder decides the local word order in a chunk. To output a target sentence, the chunk-level decoder generates a chunk representation containing global information, which the word-level decoder then uses as a basis to predict the words inside the chunk. Experimental results show that our proposed decoders can significantly improve translation performance in a WAT '16 English-to-Japanese translation task.

1 Introduction

Neural machine translation (NMT) performs end-to-end translation based on a simple encoder-decoder model (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014b) and has now overtaken the classical, complex statistical machine translation (SMT) in terms of performance and simplicity (Sennrich et al., 2016; Luong and Manning, 2016; Cromieres et al., 2016; Neubig, 2016). In NMT, an encoder first maps a source sequence into vector representations and

[∗]Contribution during internship at Microsoft Research.

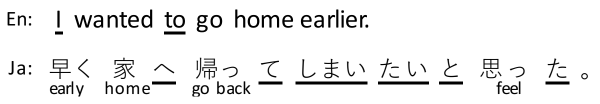

En: I wanted to go home earlier.
Ja: 早く 家 へ 帰っ て しまい たい と 思っ た 。
early
home
go back
feel

Figure 1: Translation from English to Japanese. The function words are underlined.

a decoder then maps the vectors into a target sequence (§ 2). This simple framework allows researchers to incorporate the structure of the source sentence as in SMT by leveraging various architectures as the encoder (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014b; Eriguchi et al., 2016b). Most of the NMT models, however, still rely on a sequential decoder based on a recurrent neural network (RNN) due to the difficulty in capturing the structure of a target sentence that is unseen during translation.

With the sequential decoder, however, there are two problems to be solved. First, it is difficult to model long-distance dependencies (Bahdanau et al., 2015). A hidden state h_t in an RNN is only conditioned by its previous output y_{t-1}, previous hidden state h_{t-1}, and current input x_t. This makes it difficult to capture the dependencies between an older output y_{t-N} if they are too far from the current output. This problem can become more serious when the target sequence becomes longer. For example, in Figure 1, when we translate the English sentence into the Japanese one, after the decoder predicts the content word "帰っ (go back)", it has to predict four function words "て (suffix)", "しまい (*perfect tense*)", "たい (*desire*)", and "と (to)" before predicting the next content word "思っ (feel)". In such a case, the decoder is required to capture the longer dependencies in a target sentence.

Another problem with the sequential decoder is that it is expected to cover multiple possible word orders simply by memorizing the local word se-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1901–1912
Vancouver, Canada, July 30 - August 4, 2017. ©2017 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17-1174

quences in the limited training data. This problem can be more serious in free word-order languages such as Czech, German, Japanese, and Turkish. In the case of the example in Figure 1, the order of the phrase "早く (early)" and the phrase "家へ (to home)" is flexible. This means that simply memorizing the word order in training data is not enough to train a model that can assign a high probability to a correct sentence regardless of its word order.

In the past, chunks (or phrases) were utilized to handle the above problems in statistical machine translation (SMT) (Watanabe et al., 2003; Koehn et al., 2003) and in example-based machine translation (EBMT) (Kim et al., 2010). By using a chunk rather than a word as the basic translation unit, one can treat a sentence as a shorter sequence. This makes it easy to capture the longer dependencies in a target sentence. The order of words in a chunk is relatively fixed while that in a sentence is much more flexible. Thus, modeling intra-chunk (local) word orders and inter-chunk (global) dependencies independently can help capture the difference of the flexibility between the word order and the chunk order in free word-order languages.

In this paper, we refine the original RNN decoder to consider chunk information in NMT. We propose three novel NMT models that capture and utilize the chunk structure in the target language (§ 3). Our focus is the hierarchical structure of a sentence: each sentence consists of chunks, and each chunk consists of words. To encourage an NMT model to capture the hierarchical structure, we start from a hierarchical RNN that consists of a chunk-level decoder and a word-level decoder (Model 1). Then, we improve the word-level decoder by introducing inter-chunk connections to capture the interaction between chunks (Model 2). Finally, we introduce a feedback mechanism to the chunk-level decoder to enhance the memory capacity of previous outputs (Model 3).

We evaluate the three models on the WAT '16 English-to-Japanese translation task (§ 4). The experimental results show that our best model outperforms the best single NMT model reported in WAT '16 (Eriguchi et al., 2016b).

Our contributions are twofold: (1) chunk information is introduced into NMT to improve translation performance, and (2) a novel hierarchical decoder is devised to model the properties of chunk structure in the encoder-decoder framework.

2 Preliminaries: Attention-based Neural Machine Translation

In this section, we briefly introduce the architecture of the attention-based NMT model (Bahdanau et al., 2015), which is the basis of our proposed models.

2.1 Neural Machine Translation

An NMT model usually consists of two connected neural networks: an encoder and a decoder. After the encoder maps a source sentence into a fixed-length vector, the decoder maps the vector into a target sentence. The implementation of the encoder can be a convolutional neural network (CNN) (Kalchbrenner and Blunsom, 2013), a long short-term memory (LSTM) (Sutskever et al., 2014; Luong and Manning, 2016), a gated recurrent unit (GRU) (Cho et al., 2014b; Bahdanau et al., 2015), or a Tree-LSTM (Eriguchi et al., 2016b). While various architectures are leveraged as an encoder to capture the structural information in the source language, most of the NMT models rely on a standard sequential network such as LSTM or GRU as the decoder.

Following (Bahdanau et al., 2015), we use GRU as the recurrent unit in this paper. A GRU unit computes its hidden state vector $\boldsymbol{h}_i$ given an input vector $\boldsymbol{x}_i$ and the previous hidden state $\boldsymbol{h}_{i-1}$:

$$\boldsymbol{h}_i = \mathrm{GRU}(\boldsymbol{h}_{i-1}, \boldsymbol{x}_i). \quad (1)$$

The function $\mathrm{GRU}(\cdot)$ is calculated as

$$\boldsymbol{r}_i = \sigma(\boldsymbol{W}_r \boldsymbol{x}_i + \boldsymbol{U}_r \boldsymbol{h}_{i-1} + \boldsymbol{b}_r), \quad (2)$$

$$\boldsymbol{z}_i = \sigma(\boldsymbol{W}_z \boldsymbol{x}_i + \boldsymbol{U}_z \boldsymbol{h}_{i-1} + \boldsymbol{b}_z), \quad (3)$$

$$\tilde{\boldsymbol{h}}_i = \tanh(\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{U}(\boldsymbol{r}_i \odot \boldsymbol{h}_{i-1} + \boldsymbol{b})), \quad (4)$$

$$\boldsymbol{h}_i = (\boldsymbol{1} - \boldsymbol{z}_i) \odot \tilde{\boldsymbol{h}}_i + \boldsymbol{z}_i \odot \boldsymbol{h}_{i-1}, \quad (5)$$

where vectors $\boldsymbol{r}_i$ and $\boldsymbol{z}_i$ are reset gate and update gate, respectively. While the former gate allows the model to forget the previous states, the latter gate decides how much the model updates its content. All the $\boldsymbol{W}$s and $\boldsymbol{U}$s, or the $\boldsymbol{b}$s above are trainable matrices or vectors. $\sigma(\cdot)$ and $\odot$ denote the sigmoid function and element-wise multiplication operator, respectively.

In this simple model, we train a GRU function that encodes a source sentence $\{x_1, \cdots, x_I\}$ into a single vector $\boldsymbol{h}_I$. At the same time, we jointly train another GRU function that decodes $\boldsymbol{h}_I$ to the target sentence $\{y_1, \cdots, y_J\}$. Here, the j-th word in the

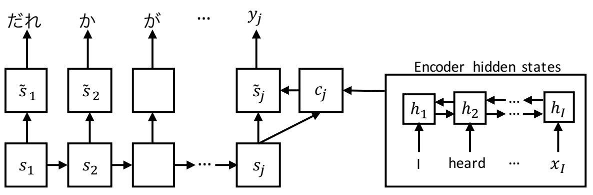


Figure 2: Standard word-based decoder.

target sentence y_j can be predicted with this decoder GRU and a nonlinear function $g(\cdot)$ followed by a softmax layer, as

$$\boldsymbol{c} = \boldsymbol{h}_I, \tag{6}$$

$$\boldsymbol{s}_j = \text{GRU}(\boldsymbol{s}_{j-1}, [\boldsymbol{y}_{j-1}; \boldsymbol{c}]), \tag{7}$$

$$\tilde{\boldsymbol{s}}_j = g(\boldsymbol{y}_{j-1}, \boldsymbol{s}_j, \boldsymbol{c}), \tag{8}$$

$$P(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) = \text{softmax}(\tilde{\boldsymbol{s}}_j), \tag{9}$$

where $\boldsymbol{c}$ is a context vector of the encoded sentence and $\boldsymbol{s}_j$ is a hidden state of the decoder GRU.

Following Bahdanau et al. (2015), we use a mini-batch stochastic gradient descent (SGD) algorithm with ADADELTA (Zeiler, 2012) to train the above two GRU functions (i.e., the encoder and the decoder) jointly. The objective is to minimize the cross-entropy loss of the training data $\boldsymbol{D}$, as

$$J = \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \boldsymbol{D}} -\log P(\boldsymbol{y}|\boldsymbol{x}). \tag{10}$$

2.2 Attention Mechanism for Neural Machine Translation

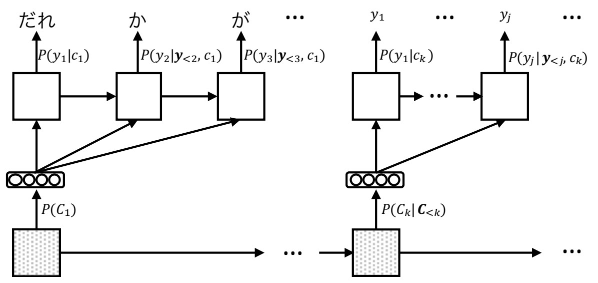
To use all the hidden states of the encoder and improve the translation performance of long sentences, Bahdanau et al. (2015) proposed using an attention mechanism. In the attention model, the context vector is not simply the last encoder state h_I but rather the weighted sum of all hidden states of the bidirectional GRU, as follows:

$$\boldsymbol{c}_j = \sum_{i=1}^{I} \alpha_{ji} \boldsymbol{h}_i. \tag{11}$$

Here, the weight α_{ji} decides how much a source word x_i contributes to the target word y_j. α_{ji} is computed by a feedforward layer and a softmax layer as

$$e_{ji} = \boldsymbol{v} \cdot \tanh(\boldsymbol{W}_e \boldsymbol{h}_i + \boldsymbol{U}_e \boldsymbol{s}_j + \boldsymbol{b}_e), \tag{12}$$

$$\alpha_{ji} = \frac{\exp(e_{ji})}{\sum_{j'=1}^{J} \exp(e_{j'i})}, \tag{13}$$



だれ
か
が
...
y_1
...
y_j
...
$P(y_1|c_1)$
$P(y_2|\boldsymbol{y}_{<2}, c_1)$
$P(y_3|\boldsymbol{y}_{<3}, c_1)$
$P(y_1|c_k)$
$P(y_j|\boldsymbol{y}_{<j}, c_k)$
$P(C_1)$
$P(C_k|\boldsymbol{C}_{<k})$

Figure 3: Chunk-based decoder. The top layer (word-level decoder) illustrates the first term in Eq. (15) and the bottom layer (chunk-level decoder) denotes the second term.

where $\boldsymbol{W}_e$, $\boldsymbol{U}_e$ are trainable matrices and the $\boldsymbol{v}$, $\boldsymbol{b}_e$ are trainable vectors.[1] In a decoder using the attention mechanism, the obtained context vector $\boldsymbol{c}_j$ in each time step replaces $\boldsymbol{c}$s in Eqs. (7) and (8). An illustration of the NMT model with the attention mechanism is shown in Figure 2.

The attention mechanism is expected to learn alignments between source and target words, and plays a similar role to the translation model in phrase-based SMT (Koehn et al., 2003).

3 Neural Machine Translation with Chunk-based Decoder

Taking non-sequential information such as chunks (or phrases) structure into consideration has proved helpful for SMT (Watanabe et al., 2003; Koehn et al., 2003) and EBMT (Kim et al., 2010). Here, we focus on two important properties of chunks (Abney, 1991): (1) The word order in a chunk is almost always fixed, and (2) A chunk consists of a few (typically one) content words surrounded by zero or more function words.

To fully utilize the above properties of a chunk, we propose modeling the intra-chunk and the inter-chunk dependencies independently with a "chunk-by-chunk" decoder (See Figure 3). In the standard word-by-word decoder described in § 2, a target word y_j in the target sentence $\boldsymbol{y}$ is predicted by taking the previous outputs $\boldsymbol{y}_{<j}$ and the source sentence $\boldsymbol{x}$ as input:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{j=1}^{J} P(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}), \tag{14}$$

where J is the length of the target sentence. Not

[1] We choose this implementation following (Luong et al., 2015b), while (Bahdanau et al., 2015) use s_{j-1} instead of s_j in Eq. (12).

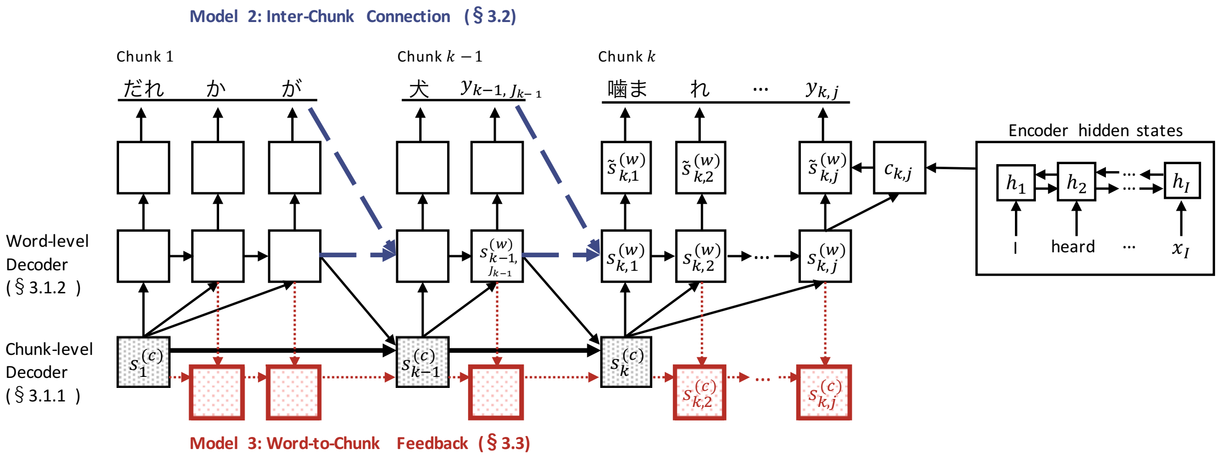


Figure 4: Proposed model: NMT with chunk-based decoder. A chunk-level decoder generates a chunk representation for each chunk while a word-level decoder uses the representation to predict each word. The solid lines in the figure illustrate Model 1. The dashed blue arrows in the word-level decoder denote the connections added in Model 2. The dotted red arrows in the chunk-level decoder denote the feedback states added in Model 3; the connections in the thick black arrows are replaced with the dotted red arrows.

assuming any structural information of the target language, the sequential decoder has to memorize long dependencies in a sequence. To release the model from the pressure of memorizing the long dependencies over a sentence, we redefine this problem as the combination of a word prediction problem and a chunk generation problem:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{k=1}^{K} \left\{ P(c_k|\boldsymbol{c}_{<k}, \boldsymbol{x}) \prod_{j=1}^{J_k} P(y_j|\boldsymbol{y}_{<j}, c_k, \boldsymbol{x}) \right\}, \tag{15}$$

where K is the number of chunks in the target sentence and J_k is the length of the k-th chunk (see Figure 3). The first term represents the generation probability of a chunk c_k and the second term indicates the probability of a word y_j in the chunk. We model the former term as a chunk-level decoder and the latter term as a word-level decoder. As demonstrated later in § 4, both K and J_k are much shorter than the sentence length J, which is why our decoders do not have to capture the long dependencies like the standard decoder does.

In the above formulation, we model the information of words and their orders in a chunk. No matter which language we target, we can assume that a chunk usually consists of some content words and function words, and the word order in the chunk is almost always fixed (Abney, 1991). Although our idea can be used in several languages, the optimal network architecture could depend on the word order of the target language. In this work, we design models for languages in which content words are followed by function words, such as Japanese and Korean. The details of our models are described in the following sections.

3.1 Model 1: Basic Chunk-based Decoder

The model described in this section is the basis of our proposed decoders. It consists of two parts: a chunk-level decoder (§ 3.1.1) and a word-level decoder (§ 3.1.2). The part drawn in black solid lines in Figure 4 illustrates the architecture of Model 1.

3.1.1 Chunk-level Decoder

Our chunk-level decoder (see Figure 3) outputs a chunk representation. The chunk representation contains the information about words that should be predicted by the word-level decoder.

To generate the representation of the k-th chunk $\tilde{\boldsymbol{s}}^{(c)}_k$, the chunk-level decoder (see the bottom layer in Figure 4) takes the last states of the word-level decoder $\boldsymbol{s}^{(w)}_{k-1,J_{k-1}}$ and updates its hidden state $\boldsymbol{s}^{(c)}_k$ as:

$$\boldsymbol{s}^{(c)}_k = \text{GRU}(\boldsymbol{s}^{(c)}_{k-1}, \boldsymbol{s}^{(w)}_{k-1,J_{k-1}}), \tag{16}$$

$$\tilde{\boldsymbol{s}}^{(c)}_k = \boldsymbol{W}_c \boldsymbol{s}^{(c)}_k + \boldsymbol{b}_c. \tag{17}$$

The obtained chunk representation $\tilde{\boldsymbol{s}}^{(c)}_k$ continues to be fed into the word-level decoder until it outputs all the words in the current chunk.

3.1.2 Word-level Decoder

Our word-level decoder (see Figure 4) differs from the standard sequential decoder described in § 2 in that it takes the chunk representation $\tilde{s}_k^{(c)}$ as input:

$$\boldsymbol{s}_{k,j}^{(w)} = \text{GRU}(\boldsymbol{s}_{k,j-1}^{(w)}, [\tilde{\boldsymbol{s}}_k^{(c)}; \boldsymbol{y}_{k,j-1}; \boldsymbol{c}_{k,j-1}]), \quad (18)$$

$$\tilde{\boldsymbol{s}}_{k,j}^{(w)} = g(\boldsymbol{y}_{k,j-1}, \boldsymbol{s}_{k,j}^{(w)}, \boldsymbol{c}_{k,j}), \quad (19)$$

$$P(y_{k,j}|\boldsymbol{y}_{<j}, \boldsymbol{x}) = \text{softmax}(\tilde{\boldsymbol{s}}_{k,j}^{(w)}). \quad (20)$$

In a standard sequential decoder, the hidden state iterates over the length of a target sentence and then generates an end-of-sentence token. In other words, its hidden layers are required to memorize the long-term dependencies and orders in the target language. In contrast, in our word-level decoder, the hidden state iterates only over the length of a chunk and then generates an end-of-chunk token. Thus, our word-level decoder is released from the pressure of memorizing the long (inter-chunk) dependencies and can focus on learning the short (intra-chunk) dependencies.

3.2 Model 2: Inter-Chunk Connection

The second term in Eq. (15) only iterates over one chunk ($j = 1$ to J_k). This means that the last state and the last output of a chunk are not being fed into the word-level decoder at the next time step (see the black part in Figure 4). In other words, $\boldsymbol{s}_{k,1}^{(w)}$ in Eq. (18) is always initialized before generating the first word in a chunk. This may have a bad influence on the word-level decoder because it cannot access any previous information at the first word of each chunk.

To address this problem, we add new connections to Model 1 between the first state in a chunk and the last state in the previous chunk, as

$$\boldsymbol{s}_{k,1}^{(w)} = \text{GRU}(\boldsymbol{s}_{k-1,J_{k-1}}^{(w)}, [\tilde{\boldsymbol{s}}_k^{(c)}; \boldsymbol{y}_{k-1,J_{k-1}}; \boldsymbol{c}_{k-1,J_{k-1}}]). \quad (21)$$

The dashed blue arrows in Figure 4 illustrate the added inter-chunk connections.

3.3 Model 3: Word-to-Chunk Feedback

The chunk-level decoder in Eq. (16) is only conditioned by $\boldsymbol{s}_{k-1,J_{k-1}}^{(w)}$, the last word state in each chunk (see the black part in Figure 4). This may affect the chunk-level decoder because it cannot memorize what kind of information has already been generated by the word-level decoder. The information about the words in a chunk should not be included in the representation of the next chunk; otherwise, it may generate the same chunks multiple times, or forget to translate some words in the source sentence.

To encourage the chunk-level decoder to memorize the information about the previous outputs more carefully, we add feedback states to our chunk-level decoder in Model 2. The feedback state in the chunk-level decoder is updated at every time step $j(> 1)$ in k-th chunk, as

$$\boldsymbol{s}_{k,j}^{(c)} = \text{GRU}(\boldsymbol{s}_{k,j-1}^{(c)}, \boldsymbol{s}_{k,j}^{(w)}). \quad (22)$$

The red part in Figure 4 illustrate the added feedback states and their connections. The connections in the thick black arrows are replaced with the dotted red arrows in Model 3.

4 Experiments

4.1 Setup

Data To examine the effectiveness of our decoders, we chose Japanese, a free word-order language, as the target language. Japanese sentences are easy to break into well-defined chunks (called *bunsetsus* (Hashimoto, 1934) in Japanese). For example, the accuracy of *bunsetsu*-chunking on newspaper articles is reported to be over 99% (Murata et al., 2000; Yoshinaga and Kitsuregawa, 2014). The effect of chunking errors in training the decoder can be suppressed, which means we can accurately evaluate the potential of our method. We used the English-Japanese training corpus in the Asian Scientific Paper Excerpt Corpus (ASPEC) (Nakazawa et al., 2016), which was provided in WAT '16. To remove inaccurate translation pairs, we extracted the first two million out of the 3 million pairs following the setting that gave the best performances in WAT '15 (Neubig et al., 2015).

Preprocessings For Japanese sentences, we performed tokenization using KyTea 0.4.7[2] (Neubig et al., 2011). Then we performed *bunsetsu*-chunking with J.DepP 2015.10.05[3] (Yoshinaga and Kitsuregawa, 2009, 2010, 2014). Special end-of-chunk tokens were inserted at the end of the chunks. Our word-level decoders described in § 3 will stop generating words after each end-of-chunk token. For English sentences, we performed the same preprocessings described on the WAT '16 Website.[4] To suppress having possible

[2]`http://www.phontron.com/kytea/`

[3]`http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/jdepp/`

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

Corpus	# words	# chunks	# sentences
Train	49,671,230	15,934,129	1,663,780
Dev.	54,287	-	1,790
Test	54,088	-	1,812

Table 1: Statistics of the target language (Japanese) in extracted corpus after preprocessing.

chunking errors affect the translation quality, we removed extremely long chunks from the training data. Specifically, among the 2 million preprocessed translation pairs, we excluded sentence pairs that matched any of following conditions: (1) The length of the source sentence or target sentence is larger than 64 (3% of whole data); (2) The maximum length of a chunk in the target sentence is larger than 8 (14% of whole data); and (3) The maximum number of chunks in the target sentence is larger than 20 (3% of whole data). Table 1 shows the details of the extracted data.

Postprocessing To perform unknown word replacement (Luong et al., 2015a), we built a bilingual English-Japanese dictionary from all of the three million translation pairs. The dictionary was extracted with the MGIZA++ 0.7.0[5] (Och and Ney, 2003; Gao and Vogel, 2008) word alignment tool by automatically extracting the alignments between English words and Japanese words.

Model Architecture Any encoder can be combined with our decoders. In this work, we adopted a single-layer bidirectional GRU (Cho et al., 2014b; Bahdanau et al., 2015) as the encoder to focus on confirming the impact of the proposed decoders. We used single layer GRUs for the word-level decoder and the chunk-level decoder. The vocabulary sizes were set to 40k for source side and 30k for target side, respectively. The conditional probability of each target word was computed with a deep-output (Pascanu et al., 2014) layer with maxout (Goodfellow et al., 2013) units following (Bahdanau et al., 2015). The maximum number of output chunks was set to 20 and the maximum length of a chunk was set to 8.

Training Details The models were optimized using ADADELTA following (Bahdanau et al., 2015). The hyperparameters of the training procedure were fixed to the values given in Table 2. Note that the learning rate was halved when the BLEU score on the development set did not increase for 30,000 batches. All the parameters were initialized randomly with Gaussian distribution. It took about a week to train each model with an NVIDIA TITAN X (Pascal) GPU.

ρ of ADADELTA	0.95
ϵ of ADADELTA	$1e^{-6}$
Initial learning rate	1.0
Gradient clipping	1.0
Mini-batch size	64
d_{hid} (dimension of hidden states)	1024
d_{emb} (dimension of word embeddings)	1024

Table 2: Hyperparameters for training.

Evaluation Following the WAT '16 evaluation procedure, we used BLEU (Papineni et al., 2002) and RIBES (Isozaki et al., 2010) to evaluate our models. The BLEU scores were calculated with `multi-bleu.pl` in Moses 2.1.1[6] (Koehn et al., 2007); RIBES scores were calculated with `RIBES.py` 1.03.1[7] (Isozaki et al., 2010). Following Cho et al. (2014a), we performed beam search[8] with length-normalized log-probability to decode target sentences. We saved the trained models that performed best on the development set during training and used them to evaluate the systems with the test set.

Baseline Systems The baseline systems and the important hyperparamters are listed in Table 3. Eriguchi et al. (2016a)'s baseline system (the first line in Table 3) was the best single (w/o ensembling) word-based NMT system that were reported in WAT '16. For a more fair evaluation, we also reimplemented a standard attention-based NMT system that uses exactly the same encoder, training procedure, and the hyperparameters as our proposed models, but has a word-based decoder. We trained this system on the training data without chunk segmentations (the second line in Table 3) and with chunk segmentations given by J.DepP (the third line in Table 3). The chunked corpus fed to the third system is exactly the same as the training data of our proposed systems (sixth to eighth lines in Table 3). In addition, we also include the Tree-to-Sequence models (Eriguchi et al., 2016a,b) (the fourth and fifth lines in Table 3) to compare the impact of capturing the structure in the source language and that in

[5]`https://github.com/moses-smt/mgiza`

[6]`http://www.statmt.org/moses/`

[7]`http://www.kecl.ntt.co.jp/icl/lirg/ribes/index.html`

[8]Beam size is set to 20.

System Encoder Type	/	Decoder Type	Hyperparameter $\lvert V_{src} \rvert$	$\lvert V_{trg} \rvert$	d_{emb}	d_{hid}	BLEU	RIBES	Dec. time [ms/sent.]
Word-based	/	Word-based (Eriguchi et al., 2016a)	88k	66k	512	512	34.64	81.60	-
	/	Word-based (our implementation)	40k	30k	1024	1024	36.33	81.22	84.1
		+ chunked training data via J.DepP	40k	30k	1024	1024	35.71	80.89	101.5
Tree-based	/	Word-based (Eriguchi et al., 2016b)	88k	66k	512	512	34.91	81.66	(363.7)[9]
	/	Char-based (Eriguchi et al., 2016a)	88k	3k	256	512	31.52	79.39	(**8.8**)[9]
Word-based	/	Proposed Chunk-based (Model 1)	40k	30k	1024	1024	34.70	81.01	165.2
		+ Inter-chunk connection (Model 2)	40k	30k	1024	1024	35.81	81.29	165.2
		+ Word-to-chunk feedback (Model 3)	40k	30k	1024	1024	**37.26**	**82.23**	163.7

Table 3: The settings and results of the baseline systems and our systems. $|V_{src}|$ and $|V_{trg}|$ denote the vocabulary size of the source language and the target language, respectively. d_{emb} and d_{hid} are the dimension size of the word embeddings and hidden states, respectively. Only single NMT models (w/o ensembling) reported in WAT '16 are listed here. Full results are available on the WAT '16 Website.[10]

the target language. Note that all systems listed in Table 3, including our models, are single models without ensemble techniques.

4.2 Results

Proposed Models vs. Baselines Table 3 shows the experimental results on the ASPEC test set. We can observe that our best model (Model 3) outperformed all the single NMT models reported in WAT '16. The gain obtained by switching Word-based decoder to Chunk-based decoder ($+0.93$ BLEU and $+1.01$ RIBES) is larger than the gain obtained by switching word-based encoder to Tree-based encoder ($+0.27$ BLEU and $+0.06$ RIBES). This result shows that capturing the chunk structure in the target language is more effective than capturing the syntax structure in the source language. Compared with the character-based NMT model (Eriguchi et al., 2016a), our Model 3 performed better by $+5.74$ BLEU score and $+2.84$ RIBES score. One possible reason for this is that using a character-based model rather than a word-based model makes it more difficult to capture long-distance dependencies because the length of a target sequence becomes much longer in the character-based model.

Comparison between Baselines Among the five baselines, our reimplementation without chunk segmentations (the second line in Table 3) achieved the best BLEU score while the Eriguchi et al. (2016b)'s system (the fourth line in Table 3) achieved the best RIBES score. The most probable reasons for the superiority of our reimplementation over the Eriguchi et al. (2016a)'s word-based baseline (the first line in Table 3) is that the dimensions of word embeddings and hidden states in our systems are higher than theirs.

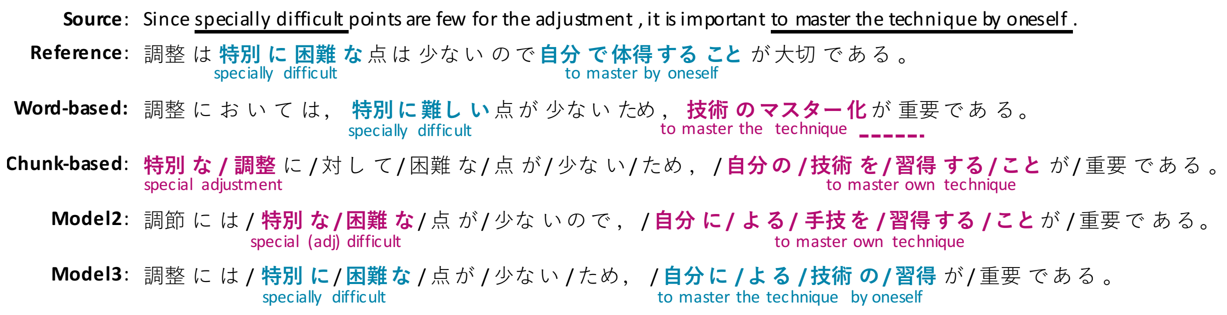
Feeding chunked training data to our baseline system (the third line in Table 3) instead of a normal data caused bad effects by -0.62 BLEU score and by -0.33 RIBES score. We evaluated the chunking ability of this system by comparing the positions of end-of-chunk tokens generated by this system with the chunk boundaries obtained by J.DepP. To our surprise, this word-based decoder could output chunk separations as accurate as our proposed Model 3 (both systems achieved F_1-score > 97). The results show that even a standard word-based decoder has the ability to predict chunk boundaries if they are given in training data. However, it is difficult for the word-based decoder to utilize the chunk information to improve the translation quality.

Decoding Speed Although the chunk-based decoder runs 2x slower than our word-based decoder, it is still practically acceptable (6 sentences per second). The character-based decoder (the fifth line in Table 3) is less time-consuming mainly because of its small vocabulary size ($|V_{trg}| = 3\text{k}$).

Chunk-level Evaluation To confirm that our models can capture local (intra-chunk) and global (inter-chunk) word orders well, we evaluated the translation quality at the chunk level. First, we performed *bunsetsu*-chunking on the reference translations in the test set. Then, for both reference translations and the outputs of our systems, we combined all the words in each chunk into a single token to regard a chunk as the basic translation unit instead of a word. Finally, we computed the chunk-based BLEU (C-BLEU) and RIBES

[9]Tree-to-Seq models are tested on CPUs instead of GPUs.

[10]`http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation`


Source: Since specially difficult points are few for the adjustment , it is important to master the technique by oneself .

Reference: 調整 は 特別 に 困難 な 点 は 少ない ので 自分 で 体得 する こと が 大切 である 。
specially difficult — to master by oneself

Word-based: 調整 に おい て は， 特別 に 難しい 点 が 少ない ため， 技術 の マスター 化 が 重要 で ある 。
specially difficult — to master the technique

Chunk-based: 特別 な / 調整 に / 対し て / 困難 な / 点 が / 少な い / ため， / 自分 の / 技術 を / 習得 する / こと が / 重要 である 。
special adjustment — to master own technique

Model2: 調節 に は / 特別 な / 困難 な / 点 が / 少な い ので， / 自分 に / よる / 手技 を / 習得 する / こと が / 重要 で ある 。
special (adj) difficult — to master own technique

Model3: 調整 に は / 特別 に / 困難 な / 点 が / 少ない / ため， / 自分 に / よる / 技術 の / 習得 が / 重要 である 。
specially difficult — to master the technique by oneself

Figure 5: Translation examples. "/" denote chunk boundaries that are automatically determined by our decoders. Words colored blue and red respectively denote correct translations and wrong translations.

Decoder	C-BLEU	C-RIBES
Word-based (our implementation)	7.56	50.73
+ chunked training data via J.DepP	7.40	51.18
Proposed Chunk-based (Model 1)	7.59	50.47
+ Inter-chunk connection (Model 2)	7.78	51.48
+ Word-to-chunk feedback (Model 3)	**8.69**	**52.82**

Table 4: Chunk-based BLEU and RIBES with the systems using the word-based encoder.

(C-RIBES). The results are listed in Table 4. For the word-based decoder (the first line in Table 4), we performed *bunsetsu*-chunking by J.DepP on its outputs to obtain chunk boundaries. As another baseline (the second line in Table 4), we used the chunked sentences as training data instead of performing chunking after decoding. The results show that our models (Model 2 and Model 3) outperform the word-based decoders in both C-BLEU and C-RIBES. This indicates that our chunk-based decoders can produce more correct chunks in a more correct order than the word-based models.

Qualitative Analysis To clarify the qualitative difference between the word-based decoder and our chunk-based decoders, we show translation examples in Figure 5. Words in blue and red respectively denote correct translations and wrong translations. The word-based decoder (our implementation) has completely dropped the translation of "by oneself." On the other hand, Model 1 generated a slightly wrong translation "自分の技術を習得すること (to master own technique)." In addition, Model 1 has made another serious word-order error "特別な調整 (special adjustment)." These results suggest that Model 1 can capture longer dependencies in a long sequence than the word-based decoder. However, Model 1 is not good at modeling global word order because it cannot access enough information about previous outputs. The weakness of modeling word order was overcome in Model 2 thanks to the inter-chunk connections. However, Model 2 still suffered from the errors of function words: it still generates a wrong chunk "特別な (special)" instead of the correct one "特別に (specially)" and a wrong chunk "よる" instead of "より." Although these errors seem trivial, such mistakes with function words bring serious changes of sentence meaning. However, all of these problems have disappeared in Model 3. This phenomenon supports the importance of the feedback states to provide the decoder with a better ability to choose more accurate words in chunks.

5 Related Work

Much work has been done on using chunk (or phrase) structure to improve machine translation quality. The most notable work involved phrase-based SMT (Koehn et al., 2003), which has been the basis for a huge amount of work on SMT for more than ten years. Apart from this, Watanabe et al. (2003) proposed a chunk-based translation model that generates output sentences in a chunk-by-chunk manner. The chunk structure is effective not only for SMT but also for example-based machine translation (EBMT). Kim et al. (2010) proposed a chunk-based EBMT and showed that using chunk structures can help with finding better word alignments. Our work is different from theirs in that our models are based on NMT, but not SMT or EBMT. The decoders in the above studies can model the chunk structure by storing chunk pairs in a large table. In contrast, we do that by individually training a chunk generation model and a word prediction model with two RNNs.

While most of the NMT models focus on the conversion between sequential data, some works have tried to incorporate non-sequential information into NMT (Eriguchi et al., 2016b; Su et al., 2017). Eriguchi et al. (2016b) use a Tree-based LSTM (Tai et al., 2015) to encode input sentence into context vectors. Given a syntactic tree of a source sentence, their tree-based encoder encodes words from the leaf nodes to the root nodes recursively. Su et al. (2017) proposed a lattice-based encoder that considers multiple tokenization results while encoding the input sentence. To prevent the tokenization errors from propagating to the whole NMT system, their attice-based encoder can utilize multiple tokenization results. These works focus on the encoding process and propose better encoders that can exploit the structures of the source language. In contrast, our work focuses on the decoding process to capture the structure of the target language. The encoders described above and our proposed decoders are complementary so they can be combined into a single network.

Considering that our Model 1 described in § 3.1 can be seen as a hierarchical RNN, our work is also related to previous studies that utilize multi-layer RNNs to capture hierarchical structures in data. Hierarchical RNNs are used for various NLP tasks such as machine translation (Luong and Manning, 2016), document modeling (Li et al., 2015; Lin et al., 2015), dialog generation (Serban et al., 2017), image captioning (Krause et al., 2016), and video captioning (Yu et al., 2016). In particular, Li et al. (2015) and Luong and Manning (2016) use hierarchical encoder-decoder models, but not for the purpose of learning syntactic structures of target sentences. Li et al. (2015) build hierarchical models at the sentence-word level to obtain better document representations. Luong and Manning (2016) build the word-character level to cope with the out-of-vocabulary problem. In contrast, we build a hierarchical models at the chunk-word level to explicitly capture the syntactic structure based on chunk segmentation.

In addition, the architecture of Model 3 is also related to stacked RNN, which has shown to be effective in improving the translation quality (Luong et al., 2015a; Sutskever et al., 2014). Although these architectures look similar to each other, there is a fundamental difference between the directions of the connection between two layers. A stacked RNN consists of multiple RNN layers that are connected from the input side to the output side at every time step. In contrast, our Model 3 has a different connection at each time step. Before it generates a chunk, there is a feed-forward connection from the chunk-level decoder to the word-level decoder. However, after generating a chunk representation, the connection is to be reversed to feed back the information from the word-level decoder to the chunk-level decoder. By switching the connections between two layers, our model can capture the chunk structure explicitly. This is the first work that proposes decoders for NMT that can capture plausible linguistic structures such as chunk.

Finally, we noticed that (Zhou et al., 2017) (which is accepted at the same time as this paper) have also proposed a chunk-based decoder for NMT. Their good experimental result on Chinese to English translation task also indicates the effectiveness of "chunk-by-chunk" decoders. Although their architecture is similar to our Model 2, there are several differences: (1) they adopt chunk-level attention instead of word-level attention; (2) their model predicts chunk tags (such as noun phrase), while ours only predicts chunk boundaries; and (3) they employ a boundary gate to decide the chunk boundaries, while we do that by simply having the model generate end-of-chunk tokens.

6 Conclusion

In this paper, we propose chunk-based decoders for NMT. As the attention mechanism in NMT plays a similar role to the translation model in phrase-based SMT, our chunk-based decoders are intended to capture the notion of chunks in chunk-based (or phrase-based) SMT. We utilize the chunk structure to efficiently capture long-distance dependencies and cope with the problem of free word-order languages such as Japanese. We designed three models that have hierarchical RNN-like architectures, each of which consists of a word-level decoder and a chunk-level decoder. We performed experiments on the WAT '16 English-to-Japanese translation task and found that our best model outperforms the strongest baselines by $+0.93$ BLEU score and by $+0.57$ RIBES score.

In future work, we will explore the optimal structures of chunk-based decoder for other free word-order languages such as Czech, German, and Turkish. In addition, we plan to combine our decoder with other encoders that capture language structure, such as a hierarchical RNN (Luong and Manning, 2016), a Tree-LSTM (Eriguchi et al., 2016b), or an order-free encoder, such as a CNN (Kalchbrenner and Blunsom, 2013).

Acknowledgements

This research was partially supported by the Research and Development on Real World Big Data Integration and Analysis program of the Ministry of Education, Culture, Sports, Science and Technology (MEXT) and RIKEN, Japan, and by the Chinese National Research Fund (NSFC) Key Project No. 61532013 and National China 973 Project No. 2015CB352401.

The authors appreciate Dongdong Zhang, Shuangzhi Wu, and Zhirui Zhang for the fruitful discussions during the first and second authors were interns at Microsoft Research Asia. We also thank Masashi Toyoda and his group for letting us use their computing resources. Finally, we thank the anonymous reviewers for their careful reading of our paper and insightful comments.

References

Steven P. Abney. 1991. Parsing by chunks. In *Principle-based parsing*, Springer, pages 257–278.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*.

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

Kyunghyun Cho, Bart van Merriënboer, 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 (EMNLP)*. pages 1724–1734.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto university participation to WAT 2016. In *Proceedings of the Third Workshop on Asian Translation (WAT)*. pages 166–174.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016a. Character-based decoding in tree-to-sequence attention-based neural machine translation. In *Proceedings of the Third Workshop on Asian Translation (WAT)*. pages 175–183.

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

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*. pages 49–57.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*. pages 1319–1327.

Shinkichi Hashimoto. 1934. *Kokugoho Yosetsu*. Meiji Shoin.

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 (EMNLP)*. pages 944–952.

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

Jae Dong Kim, Ralf D. Brown, and Jaime G. Carbonell. 2010. Chunk-based EBMT. In *Proceedings of the 14th workshop of the European Association for Machine Translation (EAMT)*.

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 (ACL)*. pages 177–180.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*. pages 48–54.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2016. A hierarchical approach for generating descriptive image paragraphs. In *arXiv:1611.06607 [cs.CV]*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. pages 1106–1115.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 899–907.

Minh-Thang Luong and Christopher D. 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 (ACL)*. pages 1054–1063.

Minh-Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015a. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the Seventh International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. pages 11–19.

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

Masaki Murata, Kiyotaka Uchimoto, Qing Ma, and Hitoshi Isahara. 2000. Bunsetsu identification using category-exclusive rules. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*. pages 565–571.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*. pages 2204–2208.

Graham Neubig. 2016. Lexicons and minimum risk training for neural machine translation: NAIST-CMU at WAT2016. In *Proceedings of the Third Workshop on Asian Translation (WAT)*. pages 119–125.

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

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 (ACL-HLT)*. pages 529–533.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51.

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

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. In *Proceedings of the Second International Conference on Learning Representations (ICLR)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation (WMT)*. pages 371–376.

Iulian V. Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NIPS)*. pages 3104–3112.

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

Taro Watanabe, Eiichiro Sumita, and Hiroshi G. Okuno. 2003. Chunk-based statistical translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 303–310.

Naoki Yoshinaga and Masaru Kitsuregawa. 2009. Polynomial to linear: Efficient classification with conjunctive features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1542–1551.

Naoki Yoshinaga and Masaru Kitsuregawa. 2010. Kernel slicing: Scalable online training with conjunctive features. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*. pages 1245–1253.

Naoki Yoshinaga and Masaru Kitsuregawa. 2014. A self-adaptive classifier for efficient text-stream processing. In *Proceedings of the 25th International*

Conference on Computational Linguistics (COLING). pages 1091–1102.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pages 4584–4593.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. In *arXiv:1212.5701 [cs.LG]*.

Hao Zhou, Zhaopeng Tu, Shujian Huang, Xiaohua Liu, Hang Li, and Jiajun Chen. 2017. Chunk-based biscale decoder for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.