Neural Machine Translation with Source-Side Latent Graph Parsing

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

Abstract

This paper presents a novel neural machine translation model which jointly learns translation and source-side latent graph representations of sentences. Unlike existing pipelined approaches using syntactic parsers, our end-to-end model learns a latent graph parser as part of the encoder of an attention-based neural machine translation model, and thus the parser is optimized according to the translation objective. In experiments, we first show that our model compares favorably with state-of-the-art sequential and pipelined syntax-based NMT models. We also show that the performance of our model can be further improved by pre-training it with a small amount of treebank annotations. Our final ensemble model significantly outperforms the previous best models on the standard English-to-Japanese translation dataset.

1 Introduction

Neural Machine Translation (NMT) is an active area of research due to its outstanding empirical results (Bahdanau et al., 2015; Luong et al., 2015; Sutskever et al., 2014). Most of the existing NMT models treat each sentence as a sequence of tokens, but recent studies suggest that syntactic information can help improve translation accuracy (Eriguchi et al., 2016b, 2017; Sennrich and Haddow, 2016; Stahlberg et al., 2016). The existing syntax-based NMT models employ a syntactic parser trained by supervised learning in advance, and hence the parser is not adapted to the translation tasks. An alternative approach for leveraging syntactic structure in a language processing task is to jointly learn syntactic trees of the sentences
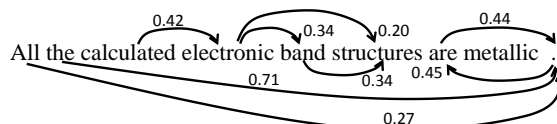

Figure 1: An example of the learned latent graphs. Edges with a small weight are omitted.

along with the target task (Socher et al., 2011; Yogatama et al., 2017).

Motivated by the promising results of recent joint learning approaches, we present a novel NMT model that can learn a task-specific latent graph structure for each source-side sentence. The graph structure is similar to the dependency structure of the sentence, but it can have cycles and is learned specifically for the translation task. Unlike the aforementioned approach of learning single syntactic trees, our latent graphs are composed of "soft" connections, i.e., the edges have real-valued weights (Figure 1). Our model consists of two parts: one is a task-independent parsing component, which we call a *latent graph parser*, and the other is an attention-based NMT model. The latent parser can be independently pre-trained with human-annotated treebanks and is then adapted to the translation task.

In experiments, we demonstrate that our model can be effectively pre-trained by the treebank annotations, outperforming a state-of-the-art sequential counterpart and a pipelined syntax-based model. Our final ensemble model outperforms the previous best results by a large margin on the WAT English-to-Japanese dataset.

2 Latent Graph Parser

We model the latent graph parser based on dependency parsing. In dependency parsing, a sentence is represented as a tree structure where each node corresponds to a word in the sentence and

a unique *root* node (ROOT) is added. Given a sentence of length N, the parent node $H_{w_i} \in \{w_1, \ldots, w_N, \text{ROOT}\}$ $(H_{w_i} \neq w_i)$ of each word w_i $(1 \leq i \leq N)$ is called its *head*. The sentence is thus represented as a set of tuples $(w_i, H_{w_i}, \ell_{w_i})$, where ℓ_{w_i} is a dependency label.

In this paper, we remove the constraint of using the tree structure and represent a sentence as a set of tuples $(w_i, p(H_{w_i}|w_i), p(\ell_{w_i}|w_i))$, where $p(H_{w_i}|w_i)$ is the probability distribution of w_i's parent nodes, and $p(\ell_{w_i}|w_i)$ is the probability distribution of the dependency labels. For example, $p(H_{w_i} = w_j|w_i)$ is the probability that w_j is the parent node of w_i. Here, we assume that a special token $\langle \text{EOS} \rangle$ is appended to the end of the sentence, and we treat the $\langle \text{EOS} \rangle$ token as ROOT. This approach is similar to that of graph-based dependency parsing (McDonald et al., 2005) in that a sentence is represented with a set of weighted arcs between the words. To obtain the *latent graph* representation of the sentence, we use a dependency parsing model based on multi-task learning proposed by Hashimoto et al. (2017).

2.1 Word Representation

The i-th input word w_i is represented with the concatenation of its d_1-dimensional word embedding $v_{dp}(w_i) \in \mathbb{R}^{d_1}$ and its character n-gram embedding $c(w_i) \in \mathbb{R}^{d_1}$: $x(w_i) = [v_{dp}(w_i); c(w_i)]$. $c(w_i)$ is computed as the average of the embeddings of the character n-grams in w_i.

2.2 POS Tagging Layer

Our latent graph parser builds upon multi-layer bi-directional Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units (Graves and Schmidhuber, 2005). In the first layer, POS tagging is handled by computing a hidden state $h_i^{(1)} = [\overrightarrow{h}_i^{(1)}; \overleftarrow{h}_i^{(1)}] \in \mathbb{R}^{2d_1}$ for w_i, where $\overrightarrow{h}_i^{(1)} = \text{LSTM}(\overrightarrow{h}_{i-1}^{(1)}, x(w_i)) \in \mathbb{R}^{d_1}$ and $\overleftarrow{h}_i^{(1)} = \text{LSTM}(\overleftarrow{h}_{i+1}^{(1)}, x(w_i)) \in \mathbb{R}^{d_1}$ are hidden states of the forward and backward LSTMs, respectively. $h_i^{(1)}$ is then fed into a softmax classifier to predict a probability distribution $p_i^{(1)} \in \mathbb{R}^{C^{(1)}}$ for word-level tags, where $C^{(1)}$ is the number of POS classes. The model parameters of this layer can be learned not only by human-annotated data, but also by backpropagation from higher layers, which are described in the next section.

2.3 Dependency Parsing Layer

Dependency parsing is performed in the second layer. A hidden state $h_i^{(2)} \in \mathbb{R}^{2d_1}$ is computed by $\overrightarrow{h}_i^{(2)} = \text{LSTM}(\overrightarrow{h}_{i-1}^{(2)}, [x(w_i); y(w_i); \overrightarrow{h}_i^{(1)}])$ and $\overleftarrow{h}_i^{(2)} = \text{LSTM}(\overleftarrow{h}_{i+1}^{(2)}, [x(w_i); y(w_i); \overleftarrow{h}_i^{(1)}])$, where $y(w_i) = W_\ell^{(1)} p_i^{(1)} \in \mathbb{R}^{d_2}$ is the POS information output from the first layer, and $W_\ell^{(1)} \in \mathbb{R}^{d_2 \times C^{(1)}}$ is a weight matrix.

Then, (soft) edges of our latent graph representation are obtained by computing the probabilities

$$p(H_{w_i} = w_j|w_i) = \frac{\exp(m(i,j))}{\sum_{k \neq i} \exp(m(i,k))}, \quad (1)$$

where $m(i,k) = h_k^{(2)\text{T}} W_{dp} h_i^{(2)}$ $(1 \leq k \leq N+1, k \neq i)$ is a scoring function with a weight matrix $W_{dp} \in \mathbb{R}^{2d_1 \times 2d_1}$. While the models of Hashimoto et al. (2017), Zhang et al. (2017), and Dozat and Manning (2017) learn the model parameters of their parsing models only by human-annotated data, we allow the model parameters to be learned by the translation task.

Next, $[h_i^{(2)}; z(H_{w_i})]$ is fed into a softmax classifier to predict the probability distribution $p(\ell_{w_i}|w_i)$, where $z(H_{w_i}) \in \mathbb{R}^{2d_1}$ is the weighted average of the hidden states of the parent nodes: $\sum_{j \neq i} p(H_{w_i} = w_j|w_i) h_j^{(2)}$. This results in the latent graph representation $(w_i, p(H_{w_i}|w_i), p(\ell_{w_i}|w_i))$ of the input sentence.

3 NMT with Latent Graph Parser

The latent graph representation described in Section 2 can be used for any sentence-level tasks, and here we apply it to an Attention-based NMT (ANMT) model (Luong et al., 2015). We modify the encoder and the decoder in the ANMT model to learn the latent graph representation.

3.1 Encoder with Dependency Composition

The ANMT model first encodes the information about the input sentence and then generates a sentence in another language. The encoder represents the word w_i with a word embedding $v_{enc}(w_i) \in \mathbb{R}^{d_3}$. It should be noted that $v_{enc}(w_i)$ is different from $v_{dp}(w_i)$ because each component is separately modeled. The encoder then takes the word embedding $v_{enc}(w_i)$ and the hidden state $h_i^{(2)}$ as the input to a uni-directional LSMT:

$$h_i^{(enc)} = \text{LSTM}(h_{i-1}^{(enc)}, [v_{enc}(w_i); h_i^{(2)}]), \quad (2)$$

where $h_i^{(enc)} \in \mathbb{R}^{d_3}$ is the hidden state corresponding to w_i. That is, the encoder of our model is a three-layer LSTM network, where the first two layers are bi-directional.

In the sequential LSTMs, relationships between words in distant positions are not *explicitly* considered. In our model, we explicitly incorporate such relationships into the encoder by defining a dependency composition function:

$$dep(w_i) = \tanh(W_{dep}[h_i^{enc}; \overline{h}(H_{w_i}); p(\ell_{w_i}|w_i)]),$$ (3)

where $\overline{h}(H_{w_i}) = \sum_{j \neq i} p(H_{w_i} = w_j|w_i)h_j^{(enc)}$ is the weighted average of the hidden states of the parent nodes.

Note on character n-gram embeddings In NMT models, sub-word units are widely used to address rare or unknown word problems (Sennrich et al., 2016). In our model, the character n-gram embeddings are fed through the latent graph parsing component. To the best of our knowledge, the character n-gram embeddings have never been used in NMT models. Wieting et al. (2016), Bojanowski et al. (2017), and Hashimoto et al. (2017) have reported that the character n-gram embeddings are useful in improving several NLP tasks by better handling unknown words.

3.2 Decoder with Attention Mechanism

The decoder of our model is a single-layer LSTM network, and the initial state is set with $h_{N+1}^{(enc)}$ and its corresponding memory cell. Given the t-th hidden state $h_t^{(dec)} \in \mathbb{R}^{d_3}$, the decoder predicts the t-th word in the target language using an attention mechanism. The attention mechanism in Luong et al. (2015) computes the weighted average of the hidden states $h_i^{(enc)}$ of the encoder:

$$s(i,t) = \frac{\exp(h_t^{(dec)} \cdot h_i^{(enc)})}{\sum_{j=1}^{N+1} \exp(h_t^{(dec)} \cdot h_j^{(enc)})},$$ (4)

$$a_t = \sum_{i=1}^{N+1} s(i,t)h_i^{(enc)},$$ (5)

where $s(i,t)$ is a scoring function which specifies how much each source-side hidden state contributes to the word prediction.

In addition, like the attention mechanism over constituency tree nodes (Eriguchi et al., 2016b), our model uses attention to the dependency composition vectors:

$$s'(i,t) = \frac{\exp(h_t^{(dec)} \cdot dep(w_i))}{\sum_{j=1}^{N} \exp(h_t^{(dec)} \cdot dep(w_j))},$$ (6)

$$a'_t = \sum_{i=1}^{N} s'(i,t)dep(w_i),$$ (7)

To predict the target word, a hidden state $\tilde{h}_t^{(dec)} \in \mathbb{R}^{d_3}$ is then computed as follows:

$$\tilde{h}_t^{(dec)} = \tanh(\tilde{W}[h_t^{(dec)}; a_t; a'_t]),$$ (8)

where $\tilde{W} \in \mathbb{R}^{d_3 \times 3d_3}$ is a weight matrix. $\tilde{h}_t^{(dec)}$ is fed into a softmax classifier to predict a target word distribution. $\tilde{h}_t^{(dec)}$ is also used in the transition of the decoder LSTMs along with a word embedding $v_{dec}(w_t) \in \mathbb{R}^{d_3}$ of the target word w_t:

$$h_{t+1}^{(dec)} = \text{LSTM}(h_t^{(dec)}, [v_{dec}(w_t); \tilde{h}_t^{(dec)}]),$$ (9)

where the use of $\tilde{h}_t^{(dec)}$ is called *input feeding* proposed by Luong et al. (2015).

The overall model parameters, including those of the latent graph parser, are jointly learned by minimizing the negative log-likelihood of the prediction probabilities of the target words in the training data. To speed up the training, we use BlackOut sampling (Ji et al., 2016). By this joint learning using Equation (3) and (7), the latent graph representations are automatically learned according to the target task.

Implementation Tips Inspired by Zoph et al. (2016), we further speed up BlackOut sampling by sharing noise samples across words in the same sentences. This technique has proven to be effective in RNN language modeling, and we have found that it is also effective in the NMT model. We have also found it effective to share the model parameters of the target word embeddings and the softmax weight matrix for word prediction (Inan et al., 2016; Press and Wolf, 2017). Also, we have found that a parameter averaging technique (Hashimoto et al., 2013) is helpful in improving translation accuracy.

Translation At test time, we use a novel beam search algorithm which combines statistics of sentence lengths (Eriguchi et al., 2016b) and length normalization (Cho et al., 2014). During the beam search step, we use the following scoring function for a generated word sequence $y = (y_1, y_2, \ldots, y_{L_y})$ given a source word sequence $x = (x_1, x_2, \ldots, x_{L_x})$:

$$\frac{1}{L_y} \left(\sum_{i=1}^{L_y} \log p(y_i|x, y_{1:i-1}) + \log p(L_y|L_x) \right),$$ (10)

where $p(L_y|L_x)$ is the probability that sentences of length L_y are generated given source-side sentences of length L_x. The statistics are taken by using the training data in advance. In our experiments, we have empirically found that this beam search algorithm helps the NMT models to avoid generating translation sentences that are too short.

4 Experimental Settings

4.1 Data

We used an English-to-Japanese translation task of the Asian Scientific Paper Excerpt Corpus (AS-PEC) (Nakazawa et al., 2016b) used in the Workshop on Asian Translation (WAT), since it has been shown that syntactic information is useful in English-to-Japanese translation (Eriguchi et al., 2016b; Neubig et al., 2015). We followed the data preprocessing instruction for the English-to-Japanese task in Eriguchi et al. (2016b). The English sentences were tokenized by the tokenizer in the Enju parser (Miyao and Tsujii, 2008), and the Japanese sentences were segmented by the KyTea tool[1]. Among the first 1,500,000 translation pairs in the training data, we selected 1,346,946 pairs where the maximum sentence length is 50. In what follows, we call this dataset the *large* training dataset. We further selected the first 20,000 and 100,000 pairs to construct the *small* and *medium* training datasets, respectively. The development data include 1,790 pairs, and the test data 1,812 pairs.

For the small and medium datasets, we built the vocabulary with words whose minimum frequency is two, and for the large dataset, we used words whose minimum frequency is three for English and five for Japanese. As a result, the vocabulary of the target language was 8,593 for the small dataset, 23,532 for the medium dataset, and 65,680 for the large dataset. A special token $\langle \text{UNK} \rangle$ was used to replace words which were not included in the vocabularies. The character n-grams ($n = 2, 3, 4$) were also constructed from each training dataset with the same frequency settings.

4.2 Parameter Optimization and Translation

We turned hyper-parameters of the model using development data. We set $(d_1, d_2) = (100, 50)$ for the latent graph parser. The word and character n-gram embeddings of the latent graph parser

were initialized with the pre-trained embeddings in Hashimoto et al. (2017).[2] The weight matrices in the latent graph parser were initialized with uniform random values in $[-\frac{\sqrt{6}}{\sqrt{row+col}}, +\frac{\sqrt{6}}{\sqrt{row+col}}]$, where row and col are the number of rows and columns of the matrices, respectively. All the bias vectors and the weight matrices in the softmax layers were initialized with zeros, and the bias vectors of the forget gates in the LSTMs were initialized by ones (Jozefowicz et al., 2015).

We set $d_3 = 128$ for the small training dataset, $d_3 = 256$ for the medium training dataset, and $d_3 = 512$ for the large training dataset. The word embeddings and the weight matrices of the NMT model were initialized with uniform random values in $[-0.1, +0.1]$. The training was performed by mini-batch stochastic gradient descent with momentum. For the BlackOut objective (Ji et al., 2016), the number of the negative samples was set to 2,000 for the small and medium training datasets, and 2,500 for the large training dataset. The mini-batch size was set to 128, and the momentum rate was set to 0.75 for the small and medium training datasets and 0.70 for the large training dataset. A gradient clipping technique was used with a clipping value of 1.0. The initial learning rate was set to 1.0, and the learning rate was halved when translation accuracy decreased. We used the BLEU scores obtained by greedy translation as the translation accuracy and checked it at every half epoch of the model training. We saved the model parameters at every half epoch and used the saved model parameters for the parameter averaging technique. For regularization, we used L2-norm regularization with a coefficient of 10^{-6} and applied dropout (Hinton et al., 2012) to Equation (8) with a dropout rate of 0.2.

The beam size for the beam search algorithm was 12 for the small and medium training datasets, and 50 for the large training dataset. We used BLEU (Papineni et al., 2002), RIBES (Isozaki et al., 2010), and perplexity scores as our evaluation metrics. Note that lower perplexity scores indicate better accuracy.

4.3 Pre-Training of Latent Graph Parser

The latent graph parser in our model can be optionally pre-trained by using human annotations for dependency parsing. In this paper we used

---

[2] The pre-trained embeddings can be found at https://github.com/hassyGo/charNgram2vec.

the widely-used Wall Street Journal (WSJ) training data to jointly train the POS tagging and dependency parsing components. We used the standard training split (Section 0-18) for POS tagging. We followed Chen and Manning (2014) to generate the training data (Section 2-21) for dependency parsing. From each training dataset, we selected the first K sentences to pre-train our model. The training dataset for POS tagging includes 38,219 sentences, and that for dependency parsing includes 39,832 sentences.

The parser including the POS tagger was first trained for 10 epochs in advance according to the multi-task learning procedure of Hashimoto et al. (2017), and then the overall NMT model was trained. When pre-training the POS tagging and dependency parsing components, we did not apply dropout to the model and did not fine-tune the word and character n-gram embeddings to avoid strong overfitting.

4.4 Model Configurations

LGP-NMT is our proposed model that learns the Latent Graph Parsing for NMT.

LGP-NMT+ is constructed by pre-training the latent parser in LGP-NMT as described in Section 4.3.

SEQ is constructed by removing the dependency composition in Equation (3), forming a sequential NMT model with the multi-layer encoder.

DEP is constructed by using pre-trained dependency relations rather than learning them. That is, $p(H_{w_i} = w_j | w_i)$ is fixed to 1.0 such that w_j is the head of w_i. The dependency labels are also given by the parser which was trained by using all the training samples for parsing and tagging.

UNI is constructed by fixing $p(H_{w_i} = w_j | w_i)$ to $\frac{1}{N}$ for all the words in the same sentence. That is, the uniform probability distributions are used for equally connecting all the words.

5 Results on Small and Medium Datasets

We first show our translation results using the small and medium training datasets. We report averaged scores with standard deviations across five different runs of the model training.

5.1 Small Training Dataset

Table 1 shows the results of using the small training dataset. LGP-NMT performs worse than SEQ

	BLEU	RIBES	Perplexity
LGP-NMT	14.31±1.49	65.96±1.86	41.13±2.66
LGP-NMT+	16.81±0.31	69.03±0.28	38.33±1.18
SEQ	15.37±1.18	67.01±1.55	38.12±2.52
UNI	15.13±1.67	66.95±1.94	39.25±2.98
DEP	13.34±0.67	64.95±0.75	43.89±1.52

Table 1: Evaluation on the development data using the small training dataset (20,000 pairs).

K	BLEU	RIBES	Perplexity
0	14.31±1.49	65.96±1.86	41.13±2.66
5,000	16.99±1.00	69.03±0.93	37.14±1.96
10,000	16.81±0.31	69.03±0.28	38.33±1.18
All	16.09±0.56	68.19±0.59	39.24±1.88

Table 2: Effects of the size K of the training datasets for POS tagging and dependency parsing.

and UNI, which shows that the small training dataset is not enough to learn useful latent graph structures from scratch. However, LGP-NMT+ ($K = 10,000$) outperforms SEQ and UNI, and the standard deviations are the smallest. Therefore, the results suggest that pre-training the parsing and tagging components can improve the translation accuracy of our proposed model. We can also see that DEP performs the worst. This is not surprising because previous studies, e.g., Li et al. (2015), have reported that using syntactic structures do not always outperform competitive sequential models in several NLP tasks.

Now that we have observed the effectiveness of pre-training our model, one question arises naturally:

> how many training samples for parsing and tagging are necessary for improving the translation accuracy?

Table 2 shows the results of using different numbers of training samples for parsing and tagging. The results of K = 0 and K = 10,000 correspond to those of LGP-NMT and LGP-NMT+ in Table 1, respectively. We can see that using the small amount of the training samples performs better than using all the training samples.[3] One possible reason is that the domains of the translation dataset and the parsing (tagging) dataset are considerably different. The parsing and tagging datasets come from WSJ, whereas the translation dataset comes from abstract text of scientific papers in a wide range of domains, such as

---

[3]We did not observe such significant difference when using the larger datasets, and we used all the training samples in the remaining part of this paper.

	BLEU	RIBES	Perplexity
LGP-NMT	28.70±0.27	77.51±0.13	12.10±0.16
LGP-NMT+	29.06±0.25	77.57±0.24	12.09±0.27
SEQ	28.60±0.24	77.39±0.15	12.15±0.12
UNI	28.25±0.35	77.13±0.20	12.37±0.08
DEP	26.83±0.38	76.05±0.22	13.33±0.23

Table 3: Evaluation on the development data using the medium training dataset (100,000 pairs).

biomedicine and computer science. These results suggest that our model can be improved by a small amount of parsing and tagging datasets in different domains. Considering the recent universal dependency project[4] which covers more than 50 languages, our model has the potential of being applied to a variety of language pairs.

5.2 Medium Training Dataset

Table 3 shows the results of using the medium training dataset. In contrast with using the small training dataset, LGP-NMT is slightly better than SEQ. LGP-NMT significantly outperforms UNI, which shows that our adaptive learning is more effective than using the uniform graph weights. By pre-training our model, LGP-NMT+ significantly outperforms SEQ in terms of the BLEU score. Again, DEP performs the worst among all the models.

By using our beam search strategy, the Brevity Penalty (BP) values of our translation results are equal to or close to 1.0, which is important when evaluating the translation results using the BLEU scores. A BP value ranges from 0.0 to 1.0, and larger values mean that the translated sentences have relevant lengths compared with the reference translations. As a result, our BLEU evaluation results are affected only by the word n-gram precision scores. BLEU scores are sensitive to the BP values, and thus our beam search strategy leads to more solid evaluation for NMT models.

6 Results on Large Dataset

Table 4 shows the BLEU and RIBES scores on the development data achieved with the large training dataset. Here we focus on our models and SEQ because UNI and DEP consistently perform worse than the other models as shown in Table 1 and 3. The averaging technique and attention-based unknown word replacement (Jean et al., 2015; Hashimoto et al., 2016) improve the scores.

B./R.	Single	+Averaging	+UnkRep
LGP-NMT	38.05/81.98	38.44/82.23	38.77/82.29
LGP-NMT+	38.75/82.13	39.01/82.40	39.37/82.48
SEQ	38.24/81.84	38.26/82.14	38.61/82.18

Table 4: BLEU (B.) and RIBES (R.) scores on the development data using the large training dataset.

	BLEU	RIBES
LGP-NMT	39.19	82.66
LGP-NMT+	39.42	82.83
SEQ	38.96	82.18
Ensemble of the above three models	41.18	83.40
Cromieres et al. (2016)	38.20	82.39
Neubig et al. (2015)	38.17	81.38
Eriguchi et al. (2016a)	36.95	82.45
Neubig and Duh (2014)	36.58	79.65
Zhu (2015)	36.21	80.91
Lee et al. (2015)	35.75	81.15

Table 5: BLEU and RIBES scores on the test data.

Again, we see that the translation scores of our model can be further improved by pre-training the model.

Table 5 shows our results on the test data, and the previous best results summarized in Nakazawa et al. (2016a) and the WAT website[5] are also shown. Our proposed models, LGP-NMT and LGP-NMT+, outperform not only SEQ but also all of the previous best results. Notice also that our implementation of the sequential model (SEQ) provides a very strong baseline, the performance of which is already comparable to the previous state of the art, even without using ensemble techniques. The confidence interval ($p \leq 0.05$) of the RIBES score of LGP-NMT+ estimated by bootstrap resampling (Noreen, 1989) is $(82.27, 83.37)$, and thus the RIBES score of LGP-NMT+ is significantly better than that of SEQ, which shows that our latent parser can be effectively pre-trained with the human-annotated treebank.

The sequential NMT model in Cromieres et al. (2016) and the tree-to-sequence NMT model in Eriguchi et al. (2016b) rely on ensemble techniques while our results mentioned above are obtained using single models. Moreover, our model is more compact[6] than the previous best NMT model in Cromieres et al. (2016). By applying the ensemble technique to LGP-NMT, LGP-NMT+,

---

[4]http://universaldependencies.org/.

[5]http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/list.php?t=1&o=1.

[6]Our training time is within five days on a c4.8xlarge machine of Amazon Web Service by our CPU-based C++ code, while it is reported that the training time is more than two weeks in Cromieres et al. (2016) by their GPU code.

Translation Example (1)
As a result , it was found that a path which <u>crosses</u> a sphere *obliquely* existed .
Reference: その結果、球内部を<u>斜めに</u><u>横切る</u>行路の 存在する ことが分かった。

LGP–NMT: その結果、球を斜めに<u>横切る</u>経路が 存在する ことが分かった。
LGP–NMT+: その結果、球を斜めに<u>横切る</u>経路が 存在する ことが分かった。
(As a result , it was found that a path which *obliquely* <u>crosses</u> a sphere existed .)

Google trans: その結果、球を<u>横切る</u>経路が斜めに 存在する ことが判明した。
SEQ: その結果、球を<u>横断</u>する経路が斜めに 存在する ことが分かった。
(As a result , it was found that a path which <u>crosses</u> a sphere existed *obliquely* .)

- -
Translation Example (2)
The androgen <u>controls</u> *negatively* ImRNA .
Reference: ImRNA はアンドロゲンにより負に 調節される 。

LGP–NMT+: アンドロゲンは ImRNA を負に 制御している 。
(The androgen *negatively* <u>controls</u> ImRNA .)

Google trans: アンドロゲンは負の ImRNA を 制御する 。
LGP–NMT: アンドロゲンは負の ImRNA を 制御する 。
(The androgen <u>controls</u> *negative* ImRNA .)

SEQ: アンドロゲンは負の ImRNA を負に 制御する 。
(The androgen *negatively* <u>controls</u> *negative* ImRNA .)

Figure 2: English-to-Japanese translation examples for focusing on the usage of adverbs.

and SEQ, the BLEU and RIBES scores are further improved, and both of the scores are significantly better than the previous best scores.

6.1 Analysis on Translation Examples

Figure 2 shows two translation examples[7] to see how the proposed model works and what is missing in the state-of-the-art sequential NMT model, SEQ. Besides the reference translation, the outputs of our models with and without pre-training, SEQ, and Google Translation[8] are shown.

Selectional Preference In the translation example (1) in Figure 2, we see that the adverb "obliquely" is interpreted differently across the systems. As in the reference translation, "obliquely" is a modifier of the verb "crosses". Our models correctly capture the relationship between the two words, whereas Google Translation and SEQ treat "obliquely" as a modifier of the verb "existed". This error is not a surprise since the verb "existed" is located closer to "obliquely" than the verb "crosses". A possible reason for the correct interpretation by our models is that they can better capture long-distance dependencies and are less susceptible to surface word distances. This is an indication of our models' ability of capturing domain-specific selectional preference that cannot be captured by purely sequential

---
[7]These English sentences were created by manual simplification of sentences in the development data.

[8]The translations were obtained at https://translate.google.com in Feb. and Mar. 2017.

models. It should be noted that simply using standard treebank-based parsers does not necessarily address this error, because our pre-trained dependency parser interprets that "obliquely" is a modifier of the verb "existed".

Adverb or Adjective The translation example (2) in Figure 2 shows another example where the adverb "negatively" is interpreted as an adverb or an adjective. As in the reference translation, "negatively" is a modifier of the verb "controls". Only LGP-NMT+ correctly captures the adverb-verb relationship, whereas "negatively" is interpreted as the adjective "negative" to modify the noun "ImRNA" in the translation results from Google Translation and LGP-NMT. SEQ interprets "negatively" as both an adverb and an adjective, which leads to the repeated translations. This error suggests that the state-of-the-art NMT models are strongly affected by the word order. By contrast, the pre-training strategy effectively embeds the information about the POS tags and the dependency relations into our model.

6.2 Analysis on Learned Latent Graphs

Without Pre-Training We inspected the latent graphs learned by LGP-NMT. Figure 1 shows an example of the learned latent graph obtained for a sentence taken from the development data of the translation task. It has long-range dependencies and cycles as well as ordinary left-to-right dependencies. We have observed that the punctuation mark "." is often pointed to by other words with large weights. This is primarily because the hidden state corresponding to the mark in each sentence has rich information about the sentence.

To measure the correlation between the latent graphs and human-defined dependencies, we parsed the sentences on the development data of the WSJ corpus and converted the graphs into dependency trees by Eisner's algorithm (Eisner, 1996). For evaluation, we followed Chen and Manning (2014) and measured Unlabeled Attachment Score (UAS). The UAS is 24.52%, which shows that the implicitly-learned latent graphs are partially consistent with the human-defined syntactic structures. Similar trends have been reported by Yogatama et al. (2017) in the case of binary constituency parsing. We checked the most dominant gold dependency labels which were assigned for the dependencies detected by LGP-NMT. The labels whose ratio is more than 3% are
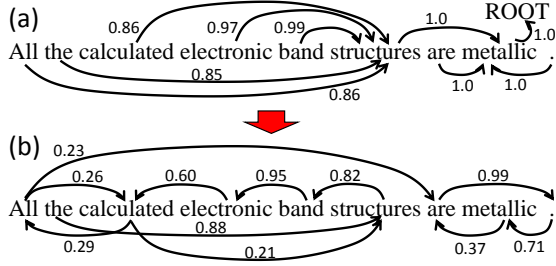
Figure 3: An example of the pre-trained dependency structures (a) and its corresponding latent graph adapted by our model (b).

nn, amod, prep, pobj, dobj, nsubj, num, det, advmod, and poss. We see that dependencies between words in distant positions, such as subject-verb-object relations, can be captured.

With Pre-Training We also inspected the pre-trained latent graphs. Figure 3-(a) shows the dependency structure output by the pre-trained latent parser for the same sentence in Figure 1. This is an ordinary dependency tree, and the head selection is almost deterministic; that is, for each word, the largest weight of the head selection is close to 1.0. By contrast, the weight values are more evenly distributed in the case of LGP-NMT as shown in Figure 1. After the overall NMT model training, the latent parser is adapted to the translation task, and Figure 3-(b) shows the adapted latent graph. Again, we can see that the adapted weight values are also distributed and different from the original pre-trained weight values, which suggests that human-defined syntax is not always optimal for the target task.

The UAS of the pre-trained dependency trees is 92.52%[9], and that of the adapted latent graphs is 18.94%. Surprisingly, the resulting UAS (18.94%) is lower than the UAS of our model without pre-training (24.52%). However, in terms of the translation accuracy, our model with pre-training is better than that without pre-training. These results suggest that human-annotated treebanks can provide useful prior knowledge to guide the overall model training by pre-training, but the resulting sentence structures adapted to the target task do not need to highly correlate with the treebanks.

---

[9]The UAS is significantly lower than the reported score in Hashimoto et al. (2017). The reason is described in Section 4.3.

7 Related Work

While initial studies on NMT treat each sentence as a sequence of words (Bahdanau et al., 2015; Luong et al., 2015; Sutskever et al., 2014), researchers have recently started investigating into the use of syntactic structures in NMT models (Bastings et al., 2017; Chen et al., 2017; Eriguchi et al., 2016a,b, 2017; Li et al., 2017; Sennrich and Haddow, 2016; Stahlberg et al., 2016; Yang et al., 2017). In particular, Eriguchi et al. (2016b) introduced a tree-to-sequence NMT model by building a tree-structured encoder on top of a standard sequential encoder, which motivated the use of the dependency composition vectors in our proposed model. Prior to the advent of NMT, the syntactic structures had been successfully used in statistical machine translation systems (Neubig and Duh, 2014; Yamada and Knight, 2001). These syntax-based approaches are pipelined; a syntactic parser is first trained by supervised learning using a treebank such as the WSJ dataset, and then the parser is used to automatically extract syntactic information for machine translation. They rely on the output from the parser, and therefore parsing errors are propagated through the whole systems. By contrast, our model allows the parser to be adapted to the translation task, thereby providing a first step towards addressing ambiguous syntactic and semantic problems, such as domain-specific selectional preference and PP attachments, in a task-oriented fashion.

Our model learns latent graph structures in a source-side language. Eriguchi et al. (2017) have proposed a model which learns to parse and translate by using automatically-parsed data. Thus, it is also an interesting direction to learn latent structures in a target-side language.

As for the learning of latent syntactic structure, there are several studies on learning task-oriented syntactic structures. Yogatama et al. (2017) used a reinforcement learning method on shift-reduce action sequences to learn task-oriented binary constituency trees. They have shown that the learned trees do not necessarily highly correlate with the human-annotated treebanks, which is consistent with our experimental results. Socher et al. (2011) used a recursive autoencoder model to greedily construct a binary constituency tree for each sentence. The autoencoder objective works as a regularization term for sentiment classification tasks. Prior to these deep learning approaches,

Wu (1997) presented a method for *bilingual parsing*. One of the characteristics of our model is directly using the soft connections of the graph edges with the real-valued weights, whereas all of the above-mentioned methods use one best structure for each sentence. Our model is based on dependency structures, and it is a promising future direction to jointly learn dependency and constituency structures in a task-oriented fashion.

Finally, more related to our model, Kim et al. (2017) applied their *structured attention networks* to a Natural Language Inference (NLI) task for learning dependency-like structures. They showed that pre-training their model by a parsing dataset did not improve accuracy on the NLI task. By contrast, our experiments show that such a parsing dataset can be effectively used to improve translation accuracy by varying the size of the dataset and by avoiding strong overfitting. Moreover, our translation examples show the concrete benefit of learning task-oriented latent graph structures.

8 Conclusion and Future Work

We have presented an end-to-end NMT model by jointly learning translation and source-side latent graph representations. By pre-training our model using treebank annotations, our model significantly outperforms both a pipelined syntax-based model and a state-of-the-art sequential model. On English-to-Japanese translation, our model outperforms the previous best models by a large margin. In future work, we investigate the effectiveness of our approach in different types of target tasks.

Acknowledgments

References

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

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. *arXiv*, cs.CL 1704.04675.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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*, pages 740–750.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. To appear.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto University Participation to WAT 2016. In *Proceedings of the 3rd Workshop on Asian Translation*, pages 166–174.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Jason Eisner. 1996. Efficient Normal-Form Parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016a. Character-based Decoding in Tree-to-Sequence Attention-based Neural Machine Translation. In *Proceedings of the 3rd Workshop on Asian Translation*, 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 (Volume 1: Long Papers)*, pages 823–833.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to Parse and Translate Improves Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. To appear.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 18(5):602–610.

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

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple Customization of Recursive Neural Networks for Semantic Relation Classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. To appear.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. *arXiv*, cs.CL 1611.01462.

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.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal Neural Machine Translation Systems for WMTf15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140.

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2016. BlackOut: Speeding up Recurrent Neural Network Language Models With Very Large Vocabularies. In *Proceedings of the 4th International Conference on Learning Representations*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2342–2350.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

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*, pages 69–73.

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.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling Source Syntax for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. To appear.

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

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98.

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

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)*.

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*.

Graham Neubig and Kevin Duh. 2014. On the Elements of an Accurate Tree-to-String Machine Translation System. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–149.

Graham Neubig, 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.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience.

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.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

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.

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 (Volume 1: Long Papers)*, pages 1715–1725.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

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 (Volume 2: Short Papers)*, pages 299–305.

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.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding Words and Sentences via Character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–404.

Kenji Yamada and Kevin Knight. 2001. A Syntax-based Statistical Translation Model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

Baosong Yang, Derek F. Wong, Tong Xiao, Lidia S. Chao, and Jingbo Zhu. 2017. Towards Bidirectional Hierarchical Representations for Attention-Based Neural Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. To appear.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to Compose Words into Sentences with Reinforcement Learning. In *Proceedings of the 5th International Conference on Learning Representations*.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency Parsing as Head Selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 665–676.

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

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.