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. Our experiments show that our model significantly outperforms the previous best results 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; Sennrich and Haddow, 2016; Stahlberg et al., 2016). The existing syntax-based NMT models employ a syntactic parser trained by supervised learning and hence rely on human-annotated treebanks. An alternative approach for leveraging syntactic structure in a language processing task is to jointly learn syntactic trees of the sentences along with the target task (Socher et al., 2011; Yogatama et al., 2016).

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. Our experiments show that our model outperforms the previous best results by a large margin on the WAT English-to-Japanese dataset.

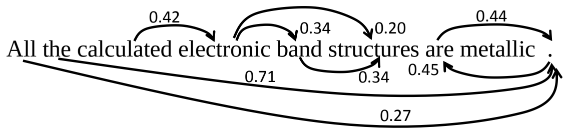
All the calculated electronic band structures are metallic .
0.42
0.34
0.20
0.44
0.71
0.34
0.45
0.27

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

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. Assuming 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 the word w_i ($1 \leq i \leq N$) is called the *head*. Then, the sentence is 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 structures and represent the 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 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 (Hashimoto et al., 2016b).

2.1 Word Representations

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

2.2 POS Tagging Layer

The 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 $\overrightarrow{h}_i^{(1)} = \mathrm{LSTM}(\overrightarrow{h}_{i-1}^{(1)}, x(w_i)) \in \mathbb{R}^{d_1}$ and $\overleftarrow{h}_i^{(1)} = \mathrm{LSTM}(\overleftarrow{h}_{i+1}^{(1)}, x(w_i)) \in \mathbb{R}^{d_1}$ 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 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

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

Then, the latent graph representation is obtained by computing the probability

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

where $m(i,k) = h_k^{(2)\mathrm{T}} W_{dp} h_i^{(2)}$ $(1 \leq k \leq N+1, k \neq i)$ is a scoring function with a weight matrix $W_{dp} \in \mathbb{R}^{2d_1 \times 2d_1}$. In the model of Hashimoto et al. (2016b) and Zhang et al. (2017), the parameters of the parsing model were learned only by human-annotated data, but 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: $\sum_{j \neq i} p(H_{w_i} = w_j|w_i) h_j^{(2)}$. Now we have 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 ideally 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: $h_i^{(enc)} = \mathrm{LSTM}(h_{i-1}^{(enc)}, [v_{enc}(w_i); h_i^{(2)}]) \in \mathbb{R}^{d_3}$. That is, the encoder of our model is a three-layer LSTM, 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:

$$dep(w_i) = \tanh(W_{dep}[h_i^{enc}; \overline{h}(H_{w_i}); p(\ell_{w_i}|w_i)]), \quad (2)$$

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.

3.2 Decoder with Attention Mechanism

The decoder of our model is a single-layer LSTM, and the initial state is initialized 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:

$$s(i,t) = \frac{\exp(h_t^{(dec)} \cdot h_i^{(enc)})}{\sum_{j=1}^{N+1} \exp(h_t^{(dec)} \cdot h_j^{(enc)})}, \quad (3)$$

$$a_t = \sum_{i=1}^{N+1} s(i,t) h_i^{(enc)}, \quad (4)$$

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

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

$$a'_t = \sum_{i=1}^{N} s'(i,t) dep(w_i), \quad (5)$$

where $s'(i,t)$ is computed in the same way as Equation (3). To predict the target word, a hidden state $\tilde{h}_t^{(dec)} \in \mathbb{R}^{d_3}$ is then computed as follows:

$$\tilde{h}_t^{(dec)} = \tanh(\tilde{W}[h_t^{(dec)}; a_t; a'_t]), \quad (6)$$

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 LSTM along with a word embedding $v_{dec}(w_t) \in \mathbb{R}^{d_3}$ of the target word w_t: $h_{t+1}^{(dec)} = \mathrm{LSTM}(h_t^{(dec)}, [v_{dec}(w_t); \tilde{h}_t^{(dec)}])$.

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 training data. To speed up the training, we use BlackOut sampling (Ji et al., 2016). By this joint learning using Equation (2) and (5), 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. Next, we have 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). Finally, we have found that a parameter averaging technique (Hashimoto et al., 2013) is helpful in improving translation accuracy.

	BLEU	RIBES	Perplexity
Our model	28.70±0.27	77.51±0.13	12.10±0.16
Pre-training	29.06±0.25	77.57±0.24	12.09±0.27
Baseline A	28.60±0.24	77.39±0.15	12.15±0.12
Baseline B	28.25±0.35	77.13±0.20	12.37±0.08

Table 1: Self comparison on the development data.

4 Experimental Settings

We briefly describe our experimental settings, and more details are available in the appendix.

4.1 Data

We used an English-to-Japanese translation task of the Asian Scientific Paper Excerpt Corpus (ASPEC) (Nakazawa et al., 2016b), since it has been shown that syntactic information is useful in English-to-Japanese translation (Eriguchi et al., 2016b; Neubig et al., 2015). We constructed a small training dataset including 100,000 translation pairs, and a large training dataset including 1,346,946 pairs. The development data includes 1,790 pairs, and the test data 1,812 pairs.

4.2 Parameter Optimization and Translation

We set $(d_1, d_2) = (100, 50)$ for the latent graph parser, $d_3 = 256$ for the small training dataset, and $d_3 = 512$ for the large training dataset. The training was performed by mini-batch stochastic gradient descent with momentum. At test time, we used a beam search algorithm with statistics of sentence lengths (Eriguchi et al., 2016b) and length normalization (Cho et al., 2014).

5 Results and Discussion

5.1 Results on Small Training Data

Table 1 shows the effects of learning the latent graph parser. Averaged scores with standard deviations across five different runs of the model training are shown. Evaluation metrics are BLEU, RIBES (Isozaki et al., 2010), and perplexity scores. Note that lower perplexity scores indicate better accuracy.

The result of "Pre-training" is obtained by pre-training the latent parser using the widely-used Wall Street Journal (WSJ) training data. We follow Chen and Manning (2014) to generate the training data for dependency parsing. The parser including the POS tagger is first trained for 10 epochs in advance according to the multi-task learning procedure of Hashimoto et al. (2016b),

B./R.	Single	+Averaging	+UnkRep
Our model	38.05/81.98	38.44/82.23	38.77/82.29
Pre-training	38.75/82.13	39.01/82.40	**39.37/82.48**
Baseline A	38.24/81.84	38.26/82.14	38.61/82.18

Table 2: BLEU (B.) and RIBES (R.) scores on the development data using the large training dataset.

	BLEU	RIBES
Our model (w/ Averaging, UnkRep)	39.19	82.66
Pre-training (w/ Averaging, UnkRep)	39.42	82.83
Baseline A (w/ Averaging, UnkRep)	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

Table 3: BLEU and RIBES scores on the test data.

and then the overall NMT model is trained. The result suggests that the pre-training can improve translation accuracy.

Baseline A is constructed by removing the dependency composition in Equation (2), forming a sequential NMT model with the multi-layer encoder. Our model is slightly better than Baseline A, but there is no significant difference. Baseline B is constructed by fixing $p(H_{w_i} = w_j|w_i)$ to $\frac{1}{N}$ for all the words in the same sentence. Our model significantly outperforms Baseline B, which shows that our adaptive learning is more effective than using the fixed graph weights.

5.2 Results on Large Training Data

Table 2 shows the BLEU and RIBES scores on the development data achieved with the large training dataset. The averaging technique and attention-based unknown word replacement (Jean et al., 2015; Hashimoto et al., 2016a) improve the scores. Note that unlike widely-used ensemble techniques, the averaging technique does not increase computational cost. In terms of the comparison between our proposed model and Baseline A, we did not observe significant difference. Again, we can see that the translation scores of our model can be further improved by pre-training the model.

Table 3 shows our results on the test data, and the previous best results summarized in Nakazawa et al. (2016a) and the WAT website[1] are also shown. Confidence intervals ($p \leq 0.05$) of the BLEU scores by bootstrap resampling (Noreen, 1989) are $(38.23, 39.98)$ for our model and $(38.43, 40.21)$ for "Pre-training". Therefore, the BLEU scores of our model with and without pre-training are significantly better than all of the previous results, but not that of Baseline A. The confidence interval of the RIBES score for "Pre-training" is $(82.27, 83.37)$, and thus the RIBES score of our pre-trained model is significantly better than that of Baseline A, which shows that our latent parser can be effectively pre-trained with the human-annotated treebank.

The NMT models in Cromieres et al. (2016) and Eriguchi et al. (2016b) rely on ensemble techniques while our results mentioned above are obtained using single models. Moreover, our model is more compact[2] than the NMT model in Cromieres et al. (2016). By applying the ensemble technique to Baseline A and our model with and without pre-training, the BLEU and RIBES scores are further improved, and both of the scores are significantly better than the previous best scores.

5.2.1 Analysis on Learned Latent Graphs

Without Pre-Training We inspected the latent graphs learned by our model without pre-training. 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. We checked the most dominant gold dependency labels which were assigned for the dependencies detected by our model. The

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

[2] 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.

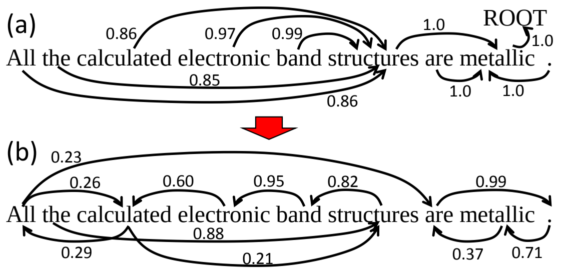
(a)
ROOT
0.86
0.97
0.99
1.0
1.0
All the calculated electronic band structures are metallic .
0.85
0.86
1.0
1.0
(b)
0.23
0.26
0.60
0.95
0.82
0.99
All the calculated electronic band structures are metallic .
0.88
0.29
0.21
0.37
0.71

Figure 2: An example of the pre-trained dependency structures (a) and its corresponding latent graph adapted by our model (b).

labels whose ratio is more than 3% are `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 2-(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 our model without pre-training as shown in Figure 1. After the overall NMT model training, the pre-trained latent parser is adapted to the translation task, and Figure 2-(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%[3], 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.

[3]The UAS is significantly lower than the reported score in Hashimoto et al. (2016b). This is because we did not apply dropout to the latent parser component and did not fine-tune the word and character n-gram embeddings during the pre-training to avoid strong overfitting.

As a result , it was found that a path which crosses a sphere *obliquely* $\boxed{\text{existed}}$.
Reference: その結果、球内部を斜めに横切る行路の$\boxed{\text{存在する}}$ことが分かった。

Our model: その結果、球を斜めに横切る経路が$\boxed{\text{存在する}}$ことが分かった。
Pre−training: その結果、球を斜めに横切る経路が$\boxed{\text{存在する}}$ことが分かった。
(As a result , it was found that a path which *obliquely* crosses a sphere $\boxed{\text{existed}}$.)

Google trans: その結果、球を横切る経路が斜めに$\boxed{\text{存在する}}$ことが判明した。
Baseline A: その結果、球を横断する経路が斜めに$\boxed{\text{存在する}}$ことが分かった。
(As a result , it was found that a path which crosses a sphere $\boxed{\text{existed}}$ *obliquely* .)

Figure 3: English-Japanese translation example.

5.2.2 Translation Example

Figure 3 shows a translation example[4]. Besides the reference translation, the outputs of our models with and without pre-training, Google Translation, and Baseline A are shown.

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 Baseline A 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 models.

6 Conclusion and Future Work

We have presented an end-to-end NMT model by jointly learning translation and source-side latent graph representations. On English-to-Japanese translation, our NMT 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

We thank Yuchen Qiao and Kenjiro Taura for their help in speeding up our training code. We also thank Akiko Eriguchi for providing the preprocessed data and helpful comments. This work was supported by CREST, JST.

[4]This English sentence was created by manual simplification of a sentence in the development data.

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*.

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.

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.

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.

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. 2016a. 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. 2016b. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. *arXiv* cs.CL 1611.01587.

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.

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.

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, 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.

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.

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.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to Compose Words into Sentences with Reinforcement Learning. *arXiv* cs.CL 1611.09100.

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*. To appear.

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.

A Supplemental Material

A.1 Data Preprocessing

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[5]. 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. This is the large training dataset, and we further sampled 100,000 pairs to construct the small training dataset. For the small dataset, 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 23,532 for the small dataset, and 65,680 for the large dataset. A special token ⟨UNK⟩ 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.

[5]http://www.phontron.com/kytea/.

A.2 Parameter Optimization

The word and character n-gram embeddings of the latent graph parser were initialized with the pre-trained embeddings in Hashimoto et al. (2016b). 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).

The word embeddings and the weight matrices of the NMT model were initialized with uniform random values in $[-0.1, +0.1]$. For training the model, the mini-batch size was set to 128, and the momentum rate was set to 0.75 for the small training dataset 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 (6) with a dropout rate of 0.8. The beam size was 12 for the small training data, and 50 for the large training data.