CUNI NMT System for WAT 2017 Translation Tasks

Tom Kocmi **Dušan Variš** **Ondřej Bojar**
Charles University,
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{kocmi,varis,bojar}@ufal.mff.cuni.cz

Abstract

The paper presents this year's CUNI submissions to the WAT 2017 Translation Task focusing on the Japanese-English translation, namely Scientific papers subtask, Patents subtask and Newswire subtask. We compare two neural network architectures, the standard sequence-to-sequence with attention (Seq2Seq) (Bahdanau et al., 2014) and an architecture using convolutional sentence encoder (FBConv2Seq) described by Gehring et al. (2017), both implemented in the NMT framework Neural Monkey[1] that we currently participate in developing. We also compare various types of preprocessing of the source Japanese sentences and their impact on the overall results. Furthermore, we include the results of our experiments with out-of-domain data obtained by combining the corpora provided for each subtask.

1 Introduction

With neural machine translation (NMT) currently becoming the leading paradigm in the field of machine translation, many novel NMT architectures with state-of-the-art results are being proposed. In the past, there were reports on large scale evaluation (Britz et al., 2017), however, the experiments were performed on a limited number of language pairs from related language families (English→German, English→French) or focused on a subset of possible NMT architectures, leaving room for further exploration.

One of the downsides of NMT is the limited allowable size of both input and output vocabularies. Various solutions for dealing with potential out-of-vocabulary (OOV) tokens were proposed either by using a back-off dictionary look-up (Luong et al., 2015), character-level translation of unknown words (Luong and Manning, 2016) or recently quite popular translation via subword units generated by byte pair encoding (Sennrich et al., 2016c). However, in the case of Japanese which has no clear definition of a word unit, there has been less research on how a particular preprocessing can influence the overall NMT performance.

In this system description paper we compare two sequence-to-sequence architectures, one using a recurrent encoder and one using a convolutional encoder. We also report results of our experiments with preprocessing of Japanese. Furthermore, we report how including additional out-of-domain training data influence the performance of NMT.

2 Dataset Preparation

In this section we describe the methods we used for preprocessing both Japanese and English.

Due to Japanese being an unsegmented language with no clear definition of word boundaries, proper text segmentation is essential. We used MeCab[2] (Kudo et al., 2004) with the UniDic[3] dictionary to perform the tokenization.

For English, we used morphological analyser MorphoDiTa[4] (Straková et al., 2014) to tokenize English training sentences. Based on the generated lemmas, we also performed truecasing of the target side of the training data.

To reduce the vocabulary size, we use byte pair encoding (BPE; Sennrich et al., 2016c) which breaks all words into subword units. The vocabulary is initialized with all alphabet characters

[1] http://ufal.mff.cuni.cz/neuralmonkey

[2] http://taku910.github.io/mecab/

[3] https://osdn.net/projects/unidic/

[4] https://github.com/ufal/morphodita/

Proceedings of the 4th Workshop on Asian Translation, pages 154–159,
Taipei, Taiwan, November 27, 2017. ©2017 AFNLP

present in the training data and larger units are added on the basis of corpus statistics. Frequent words make it to the vocabulary, less frequent words are (deterministically) broken into smaller units from the vocabulary. We generated separate BPE merges for each dataset, both source and target side.

Because the BPE algorithm, when generating the vocabulary, performs its own (subword) segmentation, we decided to compare a system trained on the tokenized Japanese (which was then further segmented by BPE) with a system that was segmented only via BPE. Additionally, we also performed a comparison with a system with Japanese text transcribed in Latin alphabet. The romanization was done by generating Hiragana transcription of each token using MeCab and then transcribing these tokens to Romaji using jaconv.[5] The resulting text was then also further segmented by BPE. The results are discussed in Section 4.1

3 Architecture Description

We use Neural Monkey[6] (Helcl and Libovický, 2017), an open-source NMT and general sequence-to-sequence learning toolkit built using the TensorFlow (Abadi et al., 2015) machine learning library.

Neural Monkey is flexible in model configuration supporting the combination of different encoder and decoder architectures as well as solving various tasks and metrics.

We perform most of the experiments on the 8GB GPU NVIDIA GeForce GTX 1080. For the preprocessing of data and final inference, we use our cluster of CPUs.

The main hyperparameters of the neural network are set as follows. We use the batch size of 60. As the optimization algorithm we use Adam (Kingma and Ba, 2014) with initial learning rate of 0.0001. We used only the non-ensembled left-to-right run (i.e. no right-to-left rescoring as done by Sennrich et al. 2016a) with beam size of 20, taking just the single-best output.

We limit the vocabulary size to 30,000 subword units. The vocabulary is constructed separately for the source and target side of the corpus.

We compare two different architectures. We describe both of them in more details as well as the hyperparameters used during the training in the following sections.

3.1 Sequence to Sequence

Our main architecture is the standard encoder-decoder architecture with attention as proposed by Bahdanau et al. (2014).

The encoder is a bidirectional recurrent neural network (BiRNN) using Gated Recurrent Units (GRU; Cho et al., 2014). In each step, it takes an embedded token from the input sequence and its previous output and outputs a representation of the token. The encoder works in both directions; the resulting vector representations at corresponding positions are concatenated. Additionally, the final outputs of both the forward and backward run are concatenated and used as the initial state of the decoder.[7]

The decoder is a standard RNN with the conditional GRU (Calixto et al., 2017) recurrent unit. At each decoding step, it takes its previous hidden state and the embedding of the token produced in the previous step as the input and produces the output vector. This vector is used to compute the attention distribution vector over the encoder outputs. The RNN output and the attention distribution vector are then used as the input of a linear layer to produce the distribution over the target vocabulary. During training, the previously generated token is replaced by the token present in the reference translation. The architecture overview is in Figure 1.

We have used the following setup of network hyperparameters. The encoder uses embeddings of size 500 and the hidden state of bidirectional GRU network of size 600 in both directions. Dropout (Srivastava et al., 2014) is turned off and the maximum length of the source sentence is set to 50 tokens. The size of the decoder hidden state is 600 and the output embedding is 500. In this case, dropout is also turned off. The maximum output length is 50 tokens. In this paper, we will refer to this architecture as *Seq2Seq*.

3.2 Convolutional Encoder

The second architecture is a hybrid system using convolutional encoder and recurrent decoder.

We use the convolutional encoder defined by Gehring et al. (2017). First, the input sequence

[5] https://github.com/ikegami-yukino/jaconv

[6] http://ufal.mff.cuni.cz/neuralmonkey

[7] The concatenated final states are transformed to match the size of the decoder hidden state.

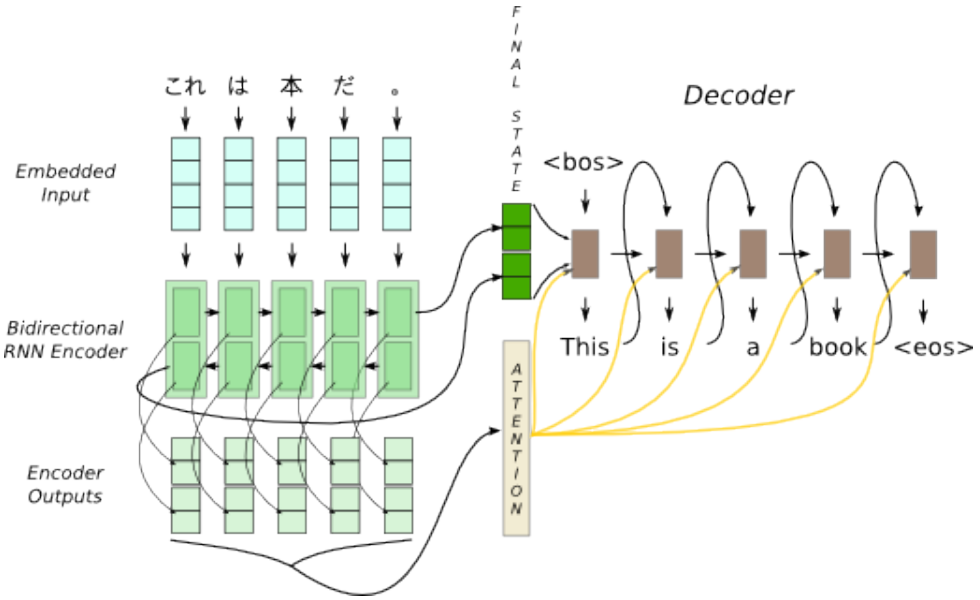

Figure 1: Simplified illustration of the standard RNN encoder-decoder architecture. Labels describing parts of the network are in italics. <*bos*> and <*eos*> are special tokens marking the beginning and end of the sentence.

of tokens $\mathbf{x} = (x_1, ..., x_n)$ is assigned a sequence of embeddings $\mathbf{w} = (w_1, ..., w_n)$ where $w_i \in \mathbb{R}^f$ is produced by embedding matrix $\mathtt{D} \in \mathbb{R}^{|V| \times f}$. When compared to the RNN encoder, the convolutional encoder does not explicitly model positions of the tokens in the input sequence. Therefore, we include this information using positional embeddings. We model the information about the position in the input sequence via $\mathbf{p} = (p_1, ..., p_n)$[8] where $p_i \in \mathbb{R}^f$. The resulting input sequence embedding is computed as $\mathbf{e} = (w_1 + p_1, ..., w_n + p_n)$.

The encoder is a convolutional network stacking several convolution blocks over each other. Each block contains a one dimensional convolution followed by a nonlinearity. The convolution with kernel size k and stride 1 with SAME padding is applied on the input sequence using d input channels and $2 \times d$ output channels. This output is then fed to the Gated Linear Unit (GLU; Dauphin et al., 2016) which substitutes a nonlinearity between the convolution blocks. Additionally, residual connections are added to the produced output. At the final layer, we get the encoded sequence $\mathbf{y} = (y_1, ..., y_n)$ where $y_i \in \mathbb{R}^d$.

We use same decoder as in the previous section. The initial decoder state $s \in \mathbb{R}^d$ is created by picking element-wise maximum across the length of the encoder output sequence $\mathbf{y}$. We tried other methods for creating the initial decoder state and this one produced the best results. Figure 2 shows the overview of the encoder architecture.

In the experiments we use encoder with the embedding size of 500 and maximum length of 50 tokens per sentence. The encoder uses 600 input features in each of its 6 convolutional layers with the kernel size of 5. Dropout is turned off. For the rest of this paper we will refer to this architecture as *FBConv2Seq*.

4 Experiments

In this section we describe all experiments we conducted for the WAT 2017 Translation Task. We report results over the development set.

4.1 Japanese Tokenization

We experimented with various tokenization methods of the Japanese source side. In Table 1 we compare untokenized, tokenized and romanized Japanese side. This experiment was evaluated over the top 1 million training examples in the ASPEC dataset.

4.2 Architecture Comparison

In Table 2 we compare the architectures we described in Section 3. We ran experiments on 4 different datasets. The JPO, JIJI, ASPEC with 1 million best sentences were used with tokenized

[8] Another option is to use the sine and cosine functions of different frequencies (Vaswani et al., 2017) instead of trainable positional embeddings.

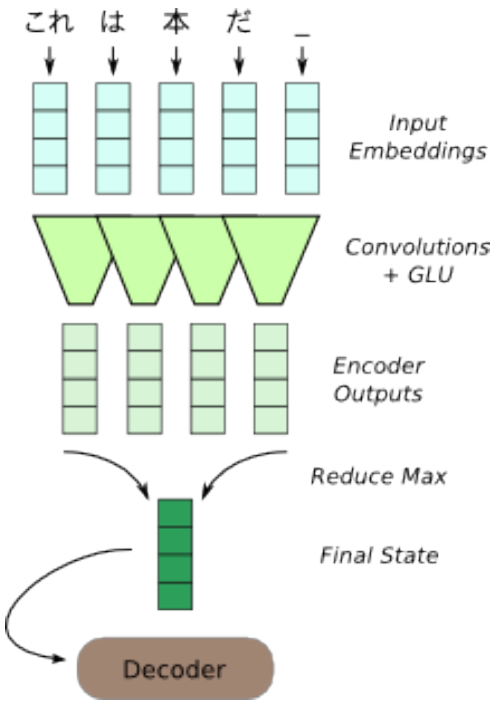

Figure 2: Illustration of the encoder used in the FBConv2Seq architecture. The attention over the *Encoder Outputs* is computed in a similar fashion as in Seq2Seq architecture and is omitted for simplicity.

Tokenization	BLEU
Untokenized	24.69
Tokenized	26.56
Romanized	25.46

Table 1: Comparison of various tokenization methods measured on the ASPEC dataset.

Japanese. The dataset ASPEC 3M was not tokenized by MeCab.

After examination of the Table 2, we can see that in most cases the Seq2Seq model (Bahdanau et al., 2014) outperforms the FBConv2Seq architecture. On the other hand, the FBConv2Seq model performed better on the untokenized corpus. This might suggest that the model has an advantage in processing inputs which are not properly segmented thanks to the convolutional nature of the encoder. This could be valuable for languages that cannot be segmented.

4.3 ASPEC Size of Data

The ASPEC dataset consists of 3 millions of English to Japanese sentence pairs ordered with a decreasing accuracy of the translation. It is a well known fact about neural networks that the more

Corpora	Seq2Seq	FBConv2Seq
JPO	**35.40 BLEU**	33.87 BLEU
JIJI	**16.40 BLEU**	13.72 BLEU
ASPEC 1M	**26.56 BLEU**	22.29 BLEU
ASPEC 3M untok.	18.14 BLEU	**19.16 BLEU**

Table 2: Comparison of two examined architectures.

Corpora	In-domain	Combined corpora
JPO (1M)	**34.95 BLEU**	33.62 BLEU
JIJI (0.2M)	**16.40 BLEU**	14.19 BLEU
ASPEC (2M)	23.19 BLEU	**23.46 BLEU**

Table 3: Comparison of in-domain data only and combined corpora.

data is available, the better performance they can get. In this experiment we try to compare the influence of the size of dataset and the quality of the training pairs. We decided to experiment with subcorpora containing 1, 1.5, 2, 2.5 and 3 million best sentence pairs. We refer to them as ASPEC 1M, ASPEC 1.5M, ASPEC 2M, ASPEC 2.5M, ASPEC 3M respectively.

For simplicity, the experiment was performed with untokenized Japanese side and we used the Seq2Seq architecture. All corpora are shuffled in order to overcome the ordering by the quality of translation.

The results presented in Figure 3 show a clear picture that the overall quality of the training data is more important than the total amount of the data.

4.4 Corpus Combination

In the previous section, we experimented with the quality of the training corpora. In this experiment we show whether more data can help in various domains or if it is also a burden as shown in the previous section comparing quality of the data.

We combined tokenized corpora for JPO (1 million sentences), JIJI (200 thousand sentences) and 2 million of the best sentences from ASPEC. The resulting corpus was shuffled.

The results in Table 3 suggest that the domain is important for both the JPO and JIJI datasets. Interestingly, it improved the score of the ASPEC 2M.

There is also another explanation which is more plausible with respect to the experiments in the previous section. The training data in JPO and JIJI have better quality than the data in ASPEC 2M, which leads to the worse performance on those datasets and on the other hand cleaner data helps

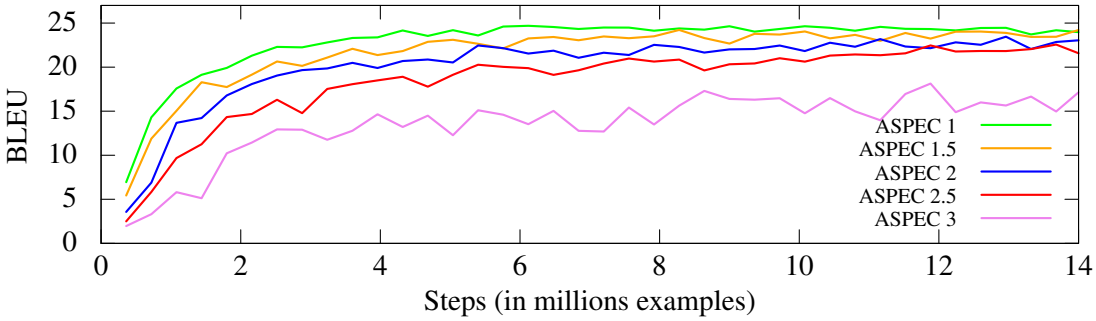

Figure 3: Learning curves over different sizes of ASPEC data.

Corpora	Results
JPO	35.35 BLEU
JIJI	16.40 BLEU
ASPEC	25.56 BLEU

Table 4: Performance of the final models on the development data.

ASPEC to increase the performance.

More research on this topic is needed to answer which of the explanations is more plausible. In future work, we want to experiment with combined corpora of JPO, JIJI and only 1 million of the cleanest sentences from ASPEC.

4.5 Official Results

Based on the previous experiments we decided to use the tokenized and shuffled in-domain training data for each of the tasks. For the Translation Task submission, we chose the Seq2Seq architecture, because it had a better overall performance. For the ASPEC dataset, we decided to train only on the 1 million cleanest training data. The results of the evaluation done on the corresponding development datasets are in Table 4.

The results of Translation Task are available on the WAT 2017 website.[9] Our system performed mostly on average. It was beaten by more sophisticated architectures using more recent state-of-the art techniques.

5 Summary

In this system description paper, we presented initial results of our research in Japanese-English NMT. We compared two different architectures implemented on NMT framework, Neural Monkey, however, as the official results of the WAT 2017 Training Task suggest, future improvements needs to be done to catch-up with the current state of the art.

We performed experiments with different input language tokenization combined with the byte-pair-encoding subword segmentation method. In the future, we plan to explore other tokenization options (e.g. splitting to bunsetsu) together with using a shared vocabulary for both the source and target languages. We are curious, whether the latter will bring an improvement when combined with romanization of Japanese.

Lastly, we made several experiments with dataset combination suggesting that including additional out-of-domain data is generally harmful for the NMT system. As the next step we plan to investigate options for creating additional synthetic data and their impact on the overall performance as suggested by Sennrich et al. (2016b).

Acknowledgments

This work has been in part supported by the European Union's Horizon 2020 research and innovation programme under grant agreements No 644402 (HimL) and 645452 (QT21), by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (projects LM2015071 and OP VVV VI CZ.02.1.01/0.0/0.0/16_013/0001781), by the Charles University Research Programme "Progres" Q18+Q48, by the Charles University SVV project number 260 453 and by the grant GAUK 8502/2016.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Cor-

[9] http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/index.html

rado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. 2017. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Doubly-attentive decoder for multi-modal neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1913–1924.

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@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning.

Jindřich Helcl and Jindřich Libovický. 2017. Neural Monkey: An Open-source Tool for Sequence Learning. *The Prague Bulletin of Mathematical Linguistics*, 107:5–17.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Proceedings of EMNLP 2004*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.

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 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

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, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*, pages 646–654, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. 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, Berlin, Germany. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Jana Straková, Milan Straka, and Jan Hajič. 2014. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.