Kyoto University Participation to WAT 2017

Fabien Cromieres
Japan Science and Technology Agency
5-3, Yonbancho, Chiyoda-ku,
Tokyo, 102-8666, Japan
fabien@pa.jst.jp

Raj Dabre
Kyoto University
Yoshida-honmachi, Sakyo-ku,
Kyoto, 606-8501, Japan
prajdabre@gmail.com

Toshiaki Nakazawa
Japan Science and Technology Agency
5-3, Yonbancho, Chiyoda-ku,
Tokyo, 102-8666, Japan
nakazawa@pa.jst.jp

Sadao Kurohashi
Kyoto University
Yoshida-honmachi, Sakyo-ku,
Kyoto, 606-8501, Japan
kuro@i.kyoto-u.ac.jp

Abstract

We describe here our approaches and results on the WAT 2017 shared translation tasks. Motivated by the good results we obtained with Neural Machine Translation in the previous shared task, we continued to explore this approach this year, with incremental improvements in models and training methods. We focused on the ASPEC dataset and could improve the state-of-the-art results for Chinese-to-Japanese and Japanese-to-Chinese translations.

1 Introduction

This paper describes our experiments for the WAT 2017 shared translation task. For more details refer to the overview paper (Nakazawa et al., 2017). This translation task contains several subtasks, but we focused on the ASPEC dataset, for the Japanese-English and Japanese-Chinese language pairs. Following up on our findings during WAT 2016 (Nakazawa et al., 2016) that our Neural Machine Translation system yielded significantly better results than our Example-Based Machine Translation system, we only experimented with NMT this year.

Our improvements are actually quite incremental, with only small changes in the model architectures, model sizes, training and decoding approaches. Together, these small changes, however, allow us to improve over our past year's results by several BLEU points, leading to the best official results for the Japanese-Chinese pair. In terms of pairwise human evaluation scores we have the best official results for all language directions except for English to Japanese. Our JPO adequacy scores are also within 1% of the best score for these language directions.

2 The Kyoto-NMT system

Following its success in the past few years, Neural Machine Translation has become the new major approach to Machine Translation. In particular, the sequence-to-sequence with attention mechanism model, first proposed in (Bahdanau et al., 2015) was proven to be very powerful and has become the de facto baseline for NMT.

Our Kyoto-NMT system largely relies on an implementation of this model, with small modifications. Kyoto-NMT is implemented using the Chainer[1] toolkit (Tokui et al., 2015). We make this implementation available under a GPL license.[2]

2.1 Overview of NMT

We describe here, briefly, our implementation based on the (Bahdanau et al., 2015) model. As shown in Figure 1, an input sentence is first converted into a sequence of vector through an embedding layer; these vectors are then fed to two LSTM layers (one going forward, the other going backward) to give a new sequence of vectors that encode the input sentence. On the decoding part of the model, a target-side sentence is generated with what is conceptually a recurrent neural network language model: an LSTM is sequentially fed the embedding of the previously generated word, and its output is sent through a deep softmax layer to produce the probability of the next word. This decoding LSTM is also fed a context vector, which is

[1] http://chainer.org/

[2] https://github.com/fabiencro/knmt . See also (Cromieres, 2016)

Proceedings of the 4th Workshop on Asian Translation, pages 146–153,
Taipei, Taiwan, November 27, 2017.

a weighted sum of the vectors encoding the input sentence, provided by the attention mechanism.

As is a common practice, we stack several layers of LSTMs for both the encoder and the decoder. When using deeper stacks of LSTMs, we can optionally add residual connections (He et al., 2016) to make the training easier. Furthermore, we also added layer normalization (He et al., 2016) to the LSTMs, which is supposed to also help training as well as regularization. However, we did not actually notice improvements when using layer normalization.

2.2 Direct connection from previous word to attention model

There is an interesting flaw in the original architecture of the model (as well as in the model described in (Bahdanau et al., 2015)). This is briefly mentioned[3] in (Goto and Tanaka, 2017), but we will expand on the details a bit more here.

The attention mechanism computes the current context using only the previous decoder state as input. But the previous decoder state has been itself computed before the previously generated target word was selected. Therefore, when computing the current context, the attention mechanism is totally unaware of the previously generated word. Intuitively, this seems wrong: the attention should certainly depend on the previously generated word.

Therefore, we add another input to the attention model: the previous word embedding. To be precise, re-using the notations from (Bahdanau et al., 2015)), the original attention is computed with this equation:

$$e_{ij} = v_a^T \tanh(W_a \cdot s_{i-1} + U_a \cdot h_j) \quad (1)$$

where e_{ij} is the unnormalized attention coefficient on source word j when decoding target word at step i, s_{i-1} is the decoder state at step $i-1$, and h_j is the encoding of source word j. The matrices W_a and U_a, and the vector v_a are the parameters of the alignment model. We replace this equation with:

$$e_{ij} = v_a^T \tanh(W_a \cdot s_{i-1} + U_a \cdot h_j + X_a \cdot E_{y-1}) \quad (2)$$

where E_{y-1} is the embedding of the previously generated target word. This increases the number of parameters by $E_o \cdot H_o$ (ie. the size of the matrix X_a), where E_o is the size of target embedings, and H_o is the size of the decoder state.

This change appeared to be remarkably efficient, giving a +1 to +2 BLEU improvement at the cost of about 1% increase in the size of the model.

2.3 Feed-Forward model

Aside from this implementation relying on LSTMs, we also implemented a model without recurrent unit but with self-attention layers, based on the model proposed in (Vaswani et al., 2017a). This model obtained state-of-the-art results on some European languages. And the code released by the author of the original paper was used as one of the organizer's baseline of WAT2017. This baseline ended up being unbeaten (in term of BLEU) by participants for the English-to-Japanese direction[4], but was inferior to other participant's submissions (including ours) in the other directions.

Our experiences with our own implementation of a feed-forward self-attention model led to results slightly inferior to the ones we obtained using a more classic LSTM-based architecture. Which is why all results presented in this paper are related to the LSTM-based model. Such feed-forward models probably have high potentials for the future, as they are more computationally efficient and do obtain state-of-the-art results on certain language directions. But, currently, we do not find that they should be necessarily preferred to recurrent architectures.

3 Models hyperparameters and pre-processing

We describe here the general settings we used for the hyperparameters of our models, as well as the pre-processing we applied to the data.

3.1 Preprocessing

As a first preprocessing step, English sentences were tokenized and lowercased. Both Japanese sentences and Chinese sentences were automatically segmented, respectively with JUMAN[5] (Kurohashi, 1994) and SKP (Shen et al., 2016).

[3] The author had previously mentioned this to us in private communications.

[4] but see section 4.2.1 for our attempt at system combination

[5] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

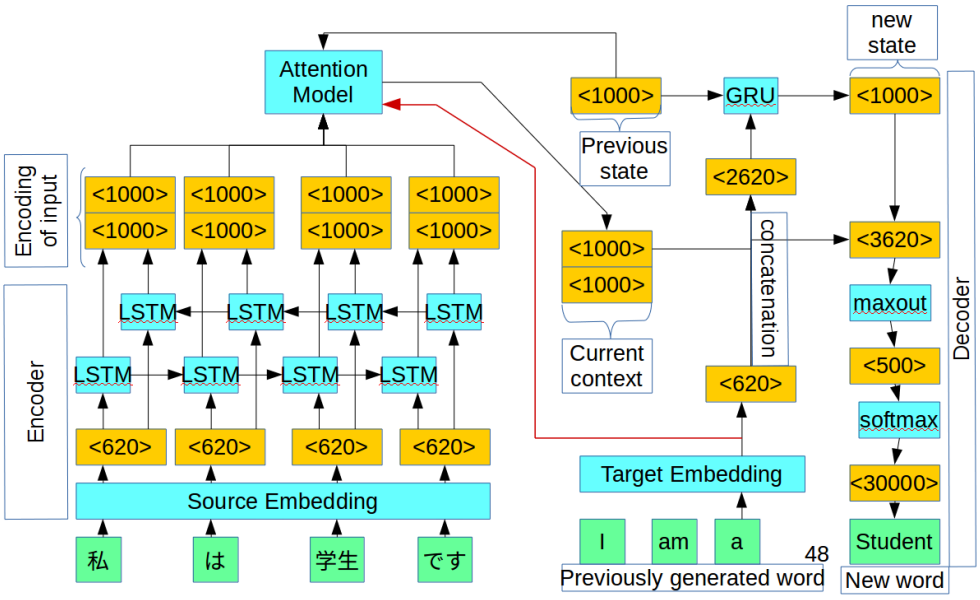

Figure 1: The structure of a NMT system with attention, as described in (Bahdanau et al., 2015) (but with LSTMs instead of GRUs). The notation "<1000>" means a vector of size 1000. The vector sizes shown here are the ones suggested in the original paper. We use this general architecture for our model, but the single LSTMs are replaced by stacks of LSTMs. We also add a connection from the target embedding to the attention model, as suggested by (Goto and Tanaka, 2017), which was not in the original model (see section 2.2)

We used subword segmentation for all target languages, so as to reduce the target vocabulary size. This makes the translation process more efficient memory-wise and computation-wise, while mostly avoiding the need for unknown-word replacement tricks such as in (Luong et al., 2015). The subword segmentation was done using the BPE algorithm (Sennrich et al., 2015) [6].

For the Japanese-Chinese language pair, we learned a joint segmentation (as suggested in (Sennrich et al., 2015)). We used a character equivalence map (Chu et al., 2013) to maximize the number of common characters between Japanese and Chinese when learning the joint segmentation. The joint segmentation was aimed at producing a vocabulary size of about 40,000 words for both the source and target vocabulary.

For the Japanese-English language pair, we did not use a joint segmentation. We created a BPE model of about 40,000 words for the target language, and about 100,000 words for the source vocabulary. Indeed, a large source vocabulary has less impact on performance than a large target vocabulary, and we expected the larger amount of data available for this language pair would let us correctly train a larger amount of embeddings.

3.2 Model hyper-parameters

For all experiments, we have used the following basic settings:

- Source and target-side embeddings of size 1024
- Source and target-side hidden states of size 1024
- Attention mechanism hidden states of size 1024
- Deep softmax output with a 2-maxout layer of size 512

We used LSTMs (Hochreiter and Schmidhuber, 1997) as the recurrent units for both the encoder and the decoder. We empirically found them to give better results than GRUs (Chung et al., 2014) in the previous shared task.

We considered stacks of 2 and 3 layers of LSTMs. Some preliminary experiments had convinced us that 4 layers or more did not lead to significant improvements, at least in the case of the

[6] using the BPE segmentation code at https://github.com/rsennrich/subword-nmt

Japanese-Chinese dataset. Adding residual connections proved to be helpful in accelerating training in the case of a 3-layers encoder-decoder[7]. They are not necessary when we stack only 2 layers of LSTMs.

We also experimented with inter-layer dropout regularization (Srivastava et al., 2014), as first suggested by (Zaremba et al., 2014).

3.3 Training Settings

Our training settings were mostly the same as those reported for WAT2016. We used ADAM (Kingma and Ba, 2014) as the training algorithm.

We also tried to do some annealing after the ADAM training. That is, we first ran ADAM until the dev loss stabilized. Then we switched to a simple stochastic gradient descent with a small learning rate ranging from 0.1 to 0.01. This process did lead to an significant further decrease of dev loss and increase of greedy dev BLEU. However, somehow surprisingly, this did not lead to a BLEU improvement when translating with the beam-search algorithm.

We used a dropout rate of 20% for the inter-layer dropout. We used L2 regularization through a weight decay factor of 1e-6. We also used an early stopping scheme: every 200 training iterations, we computed the perplexity of the development part of the ASPEC data. We also computed a BLEU score by translating this development data with a "greedy search."[8] We kept track of the parameters that gave the best development BLEU and the best development perplexity so far.

We used dynamically-sized minibatches. Minibatches were created by grouping training sentences of similar size until a threshold on the total number of words was met. The threshold was chosen so as to fill the memory of the GPU and could differ depending on the dataset and the model trained. This threshold was usually between 4000 and 8000 words per minibatch. We found these dynamically-sized minibatches to allow for faster training than the fixed-size minibatches we had used previously. We also discarded training sentences longer than 90 words.

[7] In our participation to WAT2016, we had reported having disappointing results with 3-layers encoder-decoders. We can now confirm that better results can be obtained either by a much longer training or by adding residual connections.

[8] i.e., we did not use the beam search procedure described in section 3.4, but simply translated with the most likely word at each step. Using a beam search is to slow when we need to do frequent BLEU evaluations.

As we had described in (Cromières et al., 2016), we added some additional noise to the target embeddings in the hope to make the decoder rely more on the source context than on the previously generated word when generating the next word.

3.4 Beam Search

In general, greedy decoding (that lets the decoder always select the next word with highest probability given the previously generated words) gives sub-optimal translation results. It is therefore common to use a beam-search approach to decoding, keeping a beam of translation hypotheses instead of just the greediest one.

Implementations of such a beam-search decoding can vary. We detail here the way our decoding work, which differs in some ways with, for example, the one originally provided by the LISA lab of Université de Montréal.[9]. It is an algorithm we had already used for the WAT2016 shared task and found to give good results. This time, we optionally added some more complex scoring and pruning inspired from the beam-search algorithm in (Wu et al., 2016).

We detail our basic beam search procedure in Algorithm 1. Given an input sentence i of length L_i, we first estimate the maximum length of the translation L_{mt}. L_{mt} is estimated by $L_{mt} = r \cdot L_i$, where r is a language dependent ratio. We empirically found the following values to work well : $r = 1.2$ for Japanese-to-English, $r = 2$ for English-to-Japanese, and $r = 1.5$ for Japanese-to-Chinese and Chinese-to-Japanese. At the end, we found it beneficial to rank the generated translations by their log-probability divided by their length.

Instead of our simple pruning and normalized scores, we also considered pruning and scoring functions such as the ones proposed in (Wu et al., 2016). In particular, the equation 14 of this paper describes a more complex parameterized scoring function that takes into account both the length of the translation and the coverage of the attention. We did not take the time to select the three hyper-parameters of this scoring function and just used the default ones given in the paper. As a result we could only observe benefits from this more complex scoring function for the Japanese-to-English direction (improving the results by only about 0.2 BLEU). For the three other directions, our basic

[9] https://github.com/lisa-groundhog/

algorithm gavw slightly better results. It could be that the better results could be obtained by tuning each hyperparameter to each dataset and language direction.

3.5 Averaging and Ensembling

It is well known that using an ensemble of several independently trained models can boost NMT performances by several BLEU points. We did this in the same way as was described in (Cromières et al., 2016).

On top of ensembling independently trained models, we had found it useful to also make an ensemble with the parameters of the same model corresponding respectively to the best loss, best dev BLEU and last obtained during the training process (a practice which we will call here self-ensemble). Following (Junczys-Dowmunt et al., 2016), we tried to compute averaged parameters instead of ensembling models. We found this to work surprisingly well. We observed only non-significant BLEU drops (by about 0.1 BLEU). But with the benefit that the averaged model has the same time and space complexity as a single model, while an ensemble of N models has N times the time and space complexity of a single model. We therefore switched to this averaging approach instead of the self-ensemble approach[10].

4 Results

4.1 Details for each submission

In general, all experiments were run following the methodology and hyperparameters described in section 3. We detail here the specific settings for each submissions.

Ja → En Submission 1 and 2 correspond to an ensemble of 4 models, two of them having 2 layers for encoders and decoders, and two of them having 3 layers. In submission 2, we decode using the scoring function from (Wu et al., 2016) (see section 3.4), while submission 1 uses our normal scoring function.

En → Ja Submission 1 corresponds to an ensemble of 4 models, two of them having 2 layers for encoders and decoders, and two of them having 3 layers.

[10]Of course, this is only expected to work when averaging parameters from the same training run. Ensembling remains the only option to combine independently trained models.

Ja → Zh Submission 2 corresponds to an ensemble of 5 models, three of them having 2 layers for encoders and decoders, and two of them having 3 layers. Submission 1 adds 2 additional models to the ensemble, having 3 layers on the encoder and 2 on the decoder.

Zh → Ja Submission 1 corresponds to an ensemble of 5 models, three of them having 2 layers for encoders and decoders, and two of them having 3 layers. Submission 2 does things a bit differently. It is an ensemble of 6 models using a keyword replacement method similar to (Li et al., 2016).

4.2 Official Evaluation Results

Table 1 shows the official automatic and human evaluation results of the ASPEC subtasks that we participated in. "Rank" shows the ranking of our submissions among all the submissions for each subtask.

From the point of view of human pairwise evaluation, our system achieved the best translation quality for all the subtasks except for En → Ja.

From the point of view of automatic BLEU evaluation, we obtained the best results for the two directions of the Japanese-Chinese dataset, but not for the Japanese-English dataset. In the case of JPO Adequacy scores we rank 2nd for the three language directions for which we had ranked first in term of pairwise evaluation. But because the difference in adequacy score with respect to the first system is by less than 1%, it might not be statistcially significant. For Japanese to Chinese we noticed that we had a higher percentage of translations which were rated as perfect compared to the other systems.In general the number of translations with the lowest scores (with a rating of 1) are much lower when compared to last years results which is a clear indication of progress.

It is interesting to note that these results reveal a certain discrepancy between BLEU and human evaluation. In particular, for Japanese-to-English, although our submission was significantly below some other submissions in term of BLEU, it ended up being given a higher score by human evaluation.

It somehow confirms that BLEU is not always a clear indicator of translation quality, maybe especially for a language like Japanese that has free word order. Moreover, there are questions on the reliability of BLEU when the BLEU scores are

Algorithm 1 Beam Search

Input: decoder dec conditionalized on input sentence i, beam width B
$L_{mt} \leftarrow r \cdot |i|$ ▷ L_{mt}: Maximum translation length, r: Language-dependent length ratio
$finished \leftarrow []$ ▷ list of finished translations (log-prob, translation)
$beam \leftarrow$ array of L_{mt} $item$ lists ▷ an $item$: (log-probability, decoder state, partial translation)
$beam[0] \leftarrow [(0, st_i, \text{''})]$ ▷ st_i: initial decoder state
for $n \leftarrow 1$ to L_{mt} **do**
  for $(lp, st, t) \in beam[n-1]$ **do**
    $prob, st' \leftarrow dec(st, t[-1])$ ▷ dec return the probability of next words, and the next state
    for $w, p_w \in top_B(prob)$ **do** ▷ top_B return the B words with highest probability
      if $w = EOS$ **then**
        add $(lp + log(p_w), t)$ to $finished$
      else
        add $(lp + log(p_w), st', t + w)$ to $beam[n]$
      end if
    end for
  end for
  prune $beam[n]$
end for
Sort $(lp, t) \in finished$ according to $lp/|t|$
return t s.t. $lp/|t|$ is maximum

Subtask	Ja → En		En → Ja	Ja → Zh		Zh → Ja	
Submission	1	2	1	1	2	1	2
BLEU	27.55	27.66	38.72	35.31	**35.67**	48.34	**48.43**
Rank(BLEU)	7/10	4/10	6/11	2/6	**1/6**	2/5	**1/5**
Adequacy (JPO)	4.10	-	4.26	3.95	-	4.30	-
Rank(Adequacy)	2/10*	-	4/11	2/6*	-	2/5*	-
RIBES	0.7614	0.7654	0.8324	0.8501	0.8494	**0.8842**	0.8834
AM-FM	0.5855	0.5911	0.7542	0.7854	0.7794	**0.7998**	0.7995
Human (Pairwise)	**77.75**	74.50	69.75	**72.50**	71.50	**82.75**	79.50
Rank(Human)	**1/10**	5/10	5/11	**1/6**	2/6	**1/5**	2/5

Table 1: Official automatic and human evaluation results of our NMT systems for the ASPEC subtasks. The scores in bold are the best compared to the scores of the other systems. For JPO adequacy, rank marked by a * indicates the score was within 1% of the best and therefore the difference might not be statistically significant.

very high. This hints that it might not be a good idea to use training procedures that directly optimize BLEU, something that was already mentioned in (Wu et al., 2016).

We also performed additional experiments for En → Ja after the official submission deadline which we describe in the following subsection.

4.2.1 System Combination

Considering that English-to-Japanese was the one direction where we were behind other submissions, we tried to see if we could at least get an improvement by system combination. This experience was done after the shared task results were published and is not part of the official results of WAT2017's shared task.

Tensor2Tensor's Transformer[11] (Vaswani et al., 2017b) achieved the best performance in term of BLEU (organizer's result; also the state-of-the-art) for En → Ja and we decided to combine it with our system using MEMT (Heafield and Lavie, 2010). MEMT relies on computing a lattice with various features[12] by aligning the translations at the sentence level and then using a n-gram language model for generating and ranking a n-best list. We used MEMT with the default settings which requires the following:

[11] https://github.com/tensorflow/tensor2tensor

[12] These features include paraphrases, synonyms using wordnets and common subwords using a stemmer

System	Google's Transformer	KNMT	MEMT (System Combination)
BLEU	40.79	38.74	**41.53**
RIBES	**0.8448**	0.8318	0.8410
AM-FM	0.7686	0.7565	**0.7710**

Table 2: Automatic evaluation results of system combination for English to Japanese. These results represent the SOTA in terms of BLEU and AM-FM.

- Dev set translations for both systems.
- Test set translations for both systems.
- Dev set reference sentences.
- N-gram Language Model using KenLM (We used a 6 gram model) (Heafield, 2011).

Table 2 shows the results for system combination for En → Ja. Although the Transformer model is about 2 BLEU points better than ours system combination still manages to give an increment of 0.74 BLEU which is statistically significant ($p < 0.01$).This indicates that the two models give results that are complementary. In the future we will explore methods to determine the best settings for system combination in order to further improve the translation quality.

5 Conclusion

We have detailed our methods and experimental process for our participation to the WAT2017 translation shared task. We could improve the state-of-the-art for the Japanese-Chinese dataset in term of both BLEU and pairwise human evaluation. We also obtained the best pairwise human evaluation score for Japanese-to-English translation. However, our improvements over our last year's participation were incremental and evolutionary rather than revolutionary. Small improvements across the models, training process and decoding process added up to bring a +2 to +4 BLEU improvements in the results.

In the future, we intend to do experiments with more recent evolutions of the translation models, in particular those that use more linguistic information.

Acknowledgments

This work is funded by the Japan Science and Technology Agency. We are especially grateful to Professor Isao Goto for his insightful comments on the attention model.

References

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

Chenhui Chu, Toshiaki Nakazawa, Daisuke Kawahara, and Sadao Kurohashi. 2013. Chinese-japanese machine translation exploiting chinese characters. *ACM Transactions on Asian Language Information Processing (TALIP)*, 12(4):16:1–16:25.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Fabien Cromieres. 2016. Kyoto-NMT: a neural machine translation implementation in Chainer. In *Coling 2016 System Demonstration*.

Fabien Cromières, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto university participation to wat 2016. In *Third Workshop on Asian Translation (WAT2016)*.

Isao Goto and Hideki Tanaka. 2017. Detecting untranslated content for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 47–55, Vancouver. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Kenneth Heafield and Alon Lavie. 2010. Combining machine translation output with open source: The Carnegie Mellon multi-engine machine translation scheme. *The Prague Bulletin of Mathematical Linguistics*, 93:27–36.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016. The AMU-UEDIN submission to the WMT16 news translation task: Attention-based NMT models as feature functions in phrase-based SMT. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*, pages 319–325.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sadao Kurohashi. 1994. Improvements of japanese morphological analyzer juman. In *Proceedings of the Workshop on Sharable Natural Language Resources, 1994*, pages 22–28.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. In *IJCAI*, pages 2852–2858.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL 2015*.

Toshiaki Nakazawa, Shohei Higashiyama, Chenchen Ding, Hideya Mino, Isao Goto, Graham Neubig, Hideto Kazawa, Yusuke Oda, Jun Harashima, and Sadao Kurohashi. 2017. Overview of the 4th Workshop on Asian Translation. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, Taipei, Taiwan.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Mo Shen, Li Wingmui, HyunJeong Choe, Chenhui Chu, Daisuke Kawahara, and Sadao Kurohashi. 2016. Consistent word segmentation, part-of-speech tagging and dependency labelling annotation for chinese language. In *Proceedings of the 26th International Conference on Computational Linguistics*, Osaka, Japan. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E 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(1):1929–1958.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017b. Attention is all you need. *CoRR*, abs/1706.03762.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.