Kyoto University Participation to WAT 2016

Fabien Cromieres and **Chenhui Chu** and **Toshiaki Nakazawa**
Japan Science and Technology Agency
5-3, Yonbancho, Chiyoda-ku, Tokyo, 102-8666, Japan
{fabien, chu, 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 2016 shared translation tasks. We tried to use both an example-based machine translation (MT) system and a neural MT system. We report very good translation results, especially when using neural MT for Chinese-to-Japanese translation. Overall, we could obtain best or close-to-best results in this shared task.

1 Introduction

This paper describes the experiments we did for the WAT 2016 (Nakazawa et al., 2016a) shared translation task. We used two different machine translation (MT) approaches. On one hand, we used an incremental improvement to an example-based MT (EBMT) system: the KyotoEBMT system we used for WAT 2015. On the other hand, we implemented a neural MT (NMT) system that makes use of the recent results obtained by researchers in the field. We found that the NMT approach works very well on the ASPEC (Nakazawa et al., 2016b) data, especially for the Chinese-to-Japanese direction. Overall, we could obtain the best results reported for several language directions.

This paper is decomposed as such: in Section 2, we describe the incremental improvements to our EBMT system compared with the WAT 2015 workshop. We then describe our NMT implementation and the settings we used for our experiments in Section 3. Finally, we discuss the results obtained in the shared task.

2 EBMT

The EBMT results that we submitted this year are essentially based on our KyotoEBMT system of last year (Richardson et al., 2015), but with some improvements for the data preprocessing step.

2.1 Overview of KyotoEBMT

Figure 1 shows the basic structure of the KyotoEBMT translation pipeline. The training process begins with parsing and aligning parallel sentences from the training corpus. The alignments are then used to build an example database ('translation memory') containing 'examples' or 'treelets' that form the hypotheses to be combined during decoding. Translation is performed by first parsing an input sentence then searching for treelets matching entries in the example database. The retrieved treelets are combined by a lattice-based decoder that optimizes a log linear model score. Finally, we use a reranker to select the optimal translation from the n-best list provided by the decoder using additional non-local features.[1]

2.2 Improved Data Preprocessing

English spelling correction. The English data of the ASPEC-JE task contains a lot of spelling errors. For example, the word "dielectric" is misspelled as "dieletcric" in many sentences. To address this, we apply a lattice-based correction method. We first collect correction candidates for misspelled words using the GNU Aspell toolkit.[2] In the case of "dieletcric," Aspell gives five correction candidates of "dielectric," "dielectrics,“ "dielectric's," "electric," and "dialectic." To select the correct correction for the misspelled

[1]Note that the results we submitted this year did not perform this reranking step.
[2]http://aspell.net

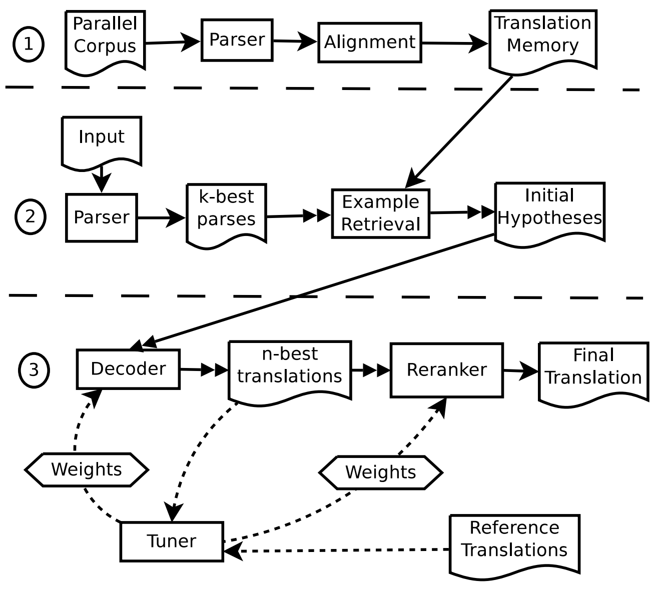

Figure 1: Overview of KyotoEBMT. The translation pipeline can be roughly divided in 3 steps. Step 1 is the creation of the example database, trained from a parallel corpus. Step 2 is the parsing of an input sentence and the generation of sets of initial hypotheses. Step 3 consists in decoding and reranking. The tuning of the weights for decoding and reranking is done by a modified version of step 3.

word, we use all the candidates given by Aspell, and compose a lattice for the sentence that contains this misspelled word. Finally, we select the best candidate based on the KyotoEBMT decoder with hand-crafted weights for three features of the edit distance between the misspelled word, the language model score, and the sentence length penalty. We verified that the correction precision of this method is about 95%. However, although over 1/3 of sentences in the training data of the ASPEC-JE task were changed after applying the spelling correction, only slight MT performance improvement was observed (about 0.1 BLEU).

Chinese short unit segmentation. For the Chinese data of the ASPEC-JC task, we applied a new segmentation standard. This standard is based on character-level POS patterns (Shen et al., 2016), which can circumvent inconsistency and address data sparsity of conventional segmentation standards. As this standard tends to segment words shorter than the conventional standards, we call it short unit segmentation. Applying short unit segmentation improves the MT performance by 0.3 BLEU on the Chinese-to-Japanese translation direction of the ASPEC-JC task.

Because of time limitation, we were not able to apply the above improved data preprocessing for our NMT system before the submission due for pairwise crowdsourcing evaluation. After the submission due, we conducted experiments using Chinese short unit segmentation on the Chinese-to-Japanese translation direction of the ASPEC-JC task, and further updated the state-of-the-art result (46.04 $\rightarrow$ 46.36 BLEU).

3 NMT

NMT is a new approach to MT that, although recently proposed, has quickly achieved state-of-the-art results (Bojar et al., 2016). We implemented our own version of the sequence-to-sequence with attention mechanism model, first proposed in (Bahdanau et al., 2015). Our implementation was done using the Chainer[3] toolkit (Tokui et al., 2015). We make this implementation available under a GPL license.[4]

[3]http://chainer.org/

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

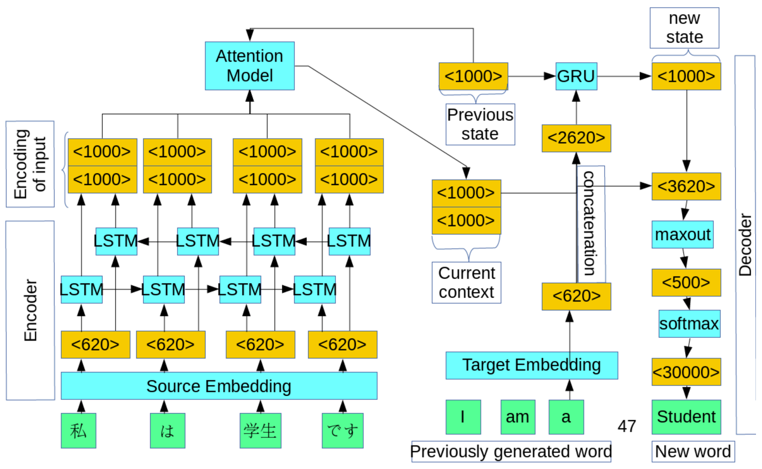

Figure 2: The structure of a NMT system with attention, as described in (Bahdanau et al., 2015) (but with LSTMs instead of GRUs). The notation "<1000>" means a vector of size 1000. The vector sizes shown here are the ones suggested in the original paper.

3.1 Overview of NMT

We describe here briefly the (Bahdanau et al., 2015) model. As shown in Figure 2, 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 a weighted sum of the vectors encoding the input sentence, provided by the attention mechanism.

3.2 Network Settings

For all experiments, we have used the following basic settings, which are mostly the same as in the original (Bahdanau et al., 2015) paper:

- Source and target-side embeddings of size 620
- Source and target-side hidden states of size 1000
- Attention mechanism hidden states of size 1000
- Deep softmax output with a 2-maxout layer of size 500

For the recurrent cells of the encoder and the decoder, we first used GRUs (Chung et al., 2014), but then switched to LSTMs (Hochreiter and Schmidhuber, 1997), as they seemed to give slightly better results. We also considered stacking several LSTMs for both the encoder and the decoder. We considered stacks of two and three LSTMs. Most of the results we provide are with stacks of two LSTMs. Although in theory we would have expected 3-layer LSTMs to performe better, we did not get better results in the few experiments we used them. One explanation is that 3-layer LSTMS are typically more difficult to train. In addition, due to time constraints, we did not take as much time as in the 2-layer setting for finding good hyperparameters for 3-layer LSTMs and training them sufficiently. In case of stacked LSTMs, we used some inter-layer dropout regularization (Srivastava et al., 2014), as first suggested by (Zaremba et al., 2014).

In addition, we have considered various vocabulary sizes. It is usually necessary to restrict the vocabulary size when using NMT systems. Especially for the target language, as the output layer will become proportionally slower to compute (and use more memory) as vocabulary size increases. We considered restricting vocabulary to 30k words and 200k words for source language. And we considered 30k and 50k words for the target languages. This means that we just keep the most frequent words, and replace less frequent words by an UNK tag.

Because of this, the network can also sometime generate UNK tags when translating an input sentence. To alleviate this issue, we use an idea first proposed in (Luong et al., 2015), which consists in replacing the generated UNK tag by a dictionary-based translation of the corresponding source word. Finding the corresponding source word is, however, not trivial. The scheme used in (Luong et al., 2015) for finding the source word does not seem like a good fit in our case, because it considers implicitly that source and target languages have similar word orders. This can be the case for English and French, but not for English and Japanese or Japanese and Chinese, as in the tasks of WAT. Therefore, we find the corresponding source word by using the attention mechanism: the word is the one on which the maximum of attention was focused on when the UNK tag was generated. The problem here is that the attention is not always focused on the correct word. This quite often leads to errors such as translating twice an input word. Still, even with this approximate unknown word replacement method, we typically get around +1$\sim$1.5 BLEU after replacement.

3.3 Training Settings

We used ADAM (Kingma and Ba, 2014) as the learning algorithm. We found it works better than ADADELTA (Zeiler, 2012) and a simple stochastic gradient descent without fine-tuning hyper-parameters. We used a dropout rate of 20% for the inter-layer dropout. We also found that using L2 regularization through weight decay was quite useful. We found that a weight decay coefficient of 1e-6 was giving optimal results for our settings. Using 1e-7 was not as effective, and using 1e-5 lead to under fitting of the model.

The training sentences were randomized, and then processed by minibatches of size 64. As processing time tends to be proportional to the length of the largest sentence in the minibatch, we used an often-used technique to make sentences in a given minibatch have similar sizes: we extract 20 x 64 sentences, sort them by size, and then create 20 minibatches from these sorted sentences. Due to memory and performance issues, it can also be necessary to limit the maximum size of training sentences. We discarded training sentences whose source or target side was larger than a given length L. Depending on the network settings, we chose L to be 80, 90 or 120.

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."[5] We kept track of the parameters that gave the best development BLEU and the best development perplexity so far. Empirically, we found that the parameter with best "greedy search" BLEU score was consistently beating the parameter with best perplexity when using beam-search. However, we could obtain even better results by ensembling these two parameters, as described in section 3.6.

3.4 Adding Noise to the Target Embeddings

We found it quite efficient to add noise to the output of the target embedding layer during training. Our rationale for doing this was that, when translating a new input, there is a risk of cascading errors: if we predict an incorrect word at time t, the embedding of this incorrect word is fed at time $t + 1$ to predict the next word, increasing the likelihood that this next word is itself incorrect. We felt that, by adding noise to the target embedding during training, we force the rest of the network to trust this input less, and therefore to be more robust to prediction errors at previous steps. The noised embeddings are created by multiplying the original embeddings with a random vector generated from a gaussian distribution with both mean and variance equal to 1.

This did appear to significantly help in our preliminary experiments, with gains ranging from +1 to +2 BLEU. However, it might be that these improvements only come from the regularization created by

[5] i.e., we did not use the beam search procedure described in section 3.5, 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.

the noise, and is not directly linked to an increased robustness to prediction errors. We should do further experiments to verify if there is a marked difference in results between adding noise the way we do, and simply adding a dropout layer to any part of the network.

3.5 Beam Search

The naive way to translate using an NMT system such as the one we described is, after feeding the input sentence, to produce the target words with the highest probability at each step. This is however sub-optimal, as a better translation might include a sub-sequence that is locally less likely than another. Hence, the idea is to use a form of beam-search to keep track of several translation hypotheses at once.

There are a few differences in the way we handle beam search, compared with other NMT implementations such as the one originally provided by the LISA lab of Université de Montréal.[6] We came to this method after a few iterative trials. We detail our 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 simply using the log-probability. This helps to counter-balance the fact that the model will otherwise tend to be biased towards shorter translations. One could fear that doing this will actually bias the model towards longer translations, but we did not observe such a thing; maybe in part due to the fact that our algorithm caps the maximum length of a translation through the language-dependent length ratio.

Although we have seen some claims that large beam width was not very helpful for NMT decoding, we actually verified empirically that using a beam-width of 100 could give significantly better results than a beam-width of 30 or less (of course, at the cost of a serious slow-down in the translation speed). We used a beam-width of 100 in all our submitted results.

Algorithm 1 Beam Search

```
1: Input: decoder $dec$ conditionalized on input sentence $i$, beam width $B$
2: $L_{mt} \leftarrow r \cdot |i|$    ▷ $L_{mt}$: Maximum translation length, $r$: Language-dependent length ratio
3: $finished \leftarrow []$    ▷ list of finished translations (log-prob, translation)
4: $beam \leftarrow$ array of $L_{mt}$ $item$ lists    ▷ an $item$: (log-probability, decoder state, partial translation)
5: $beam[0] \leftarrow [(0, st_i, ”)]$    ▷ $st_i$: initial decoder state
6: for $n \leftarrow 1$ to $L_{mt}$ do
7:     for $(lp, st, t) \in beam[n-1]$ do
8:         $prob, st' \leftarrow dec(st, t[-1])$    ▷ $dec$ return the probability of next words, and the next state
9:         for $w, p_w \in top_B(prob)$ do    ▷ $top_B$ return the $B$ words with highest probability
10:            if $w = EOS$ then
11:                add $(lp + log(p_w), t)$ to $finished$
12:            else
13:                add $(lp + log(p_w), st', t + w)$ to $beam[n]$
14:            end if
15:        end for
16:    end for
17:    prune $beam[n]$
18: end for
19: Sort $(lp, t) \in finished$ according to $lp/|t|$
20: return $t$ s.t. $lp/|t|$ is maximum
```

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

3.6 Ensembling

Ensembling has previously been found to widely increase translation quality of NMT systems. Ensembling essentially means that, at each translation step, we predict the next word using several NMT models instead of one. The two "classic" ways of combining the prediction of different systems are to either take the geometric average or the arithmetic average of their predicted probabilities. Interestingly, although it seems other researchers have reported that using the arithmetic average works better (Luong et al., 2016), we actually found that geometric average was giving better results for us.

Ensembling usually works best with independently trained parameters. We actually found that even using parameters from a single run could improve results. This had also been previously observed by (Sennrich et al., 2016). Therefore, for the cases when we could only run one training session, we ensembled the three parameters corresponding to the best development loss, the best development BLEU, and the final parameters (obtained after continuing training for some time after the best development BLEU was found). We refer to this as "self-ensembling" in the result section. When we could do n independent training, we kept these three parameters for each of the independent run and ensembled the $3 \cdot n$ parameters.

3.7 Preprocessing

The preprocessing steps were essentially the same as for KyotoEBMT. We lowercased the English sentences, and segmented automatically the Japanese and Chinese sentences. We used JUMAN to segment the Japanese and SKP to segment the Chinese. We also tried to apply BPE segmentation (Sennrich et al., 2015) in some cases.[7]

4 Results

We submitted the translation results of both EBMT and NMT systems, however, the automatic evaluation results of NMT seemed to be quite superior to those of EBMT. Therefore, we selected NMT results for the human evaluations for almost all the subtasks.

4.1 Specifics NMT Settings for Each Language Pair

The NMT training is quite slow. On an Nvidia Titan X (Maxwell), we typically needed 4∼7 days of training for ASPEC-CJ, and more than 2 weeks for ASPEC-EJ. Therefore, it took us a lot of time to experiment with different variations of settings, and we did not have the time to fully re-run experiments with the best settings. We describe here the settings of the results we submitted for human evaluation.

Ja → En Submission 1 corresponds to only one trained model, with self-ensembing (see Section 3.6). In this case, source vocabulary was reduced to 200k words. For target vocabulary, we reduced the vocabulary to 52k using BPE. We used a double-layer LSTM.

Submission 2 is an ensemble of four independently trained models, two of which were using GRUs, and two of which were using LSTM. Source and target vocabularies were restricted to 30k words. For each model, we actually used three sets of parameters (best development BLEU, best development perplexity, final), as described in Section 3.6. Therefore, ensembling was actually using 12 different models.

En → Ja Submission 1 also corresponds to only one trained model. BPE was applied to both source and target sides to reduce vocabulary to 52k. We used a double-layer LSTM and self-ensembling.

Ja → Zh Submission 1 corresponds to one trained model with self-ensembling, with double-layer LSTM and 30k words for both source and target vocabularies.

Zh → Ja Submission 1 corresponds to ensembling two independently trained models using double-layer LSTM with 30k source and target vocabularies. In addition, a model was trained on reversed Japanese sentences and was used to rescore the translations.

Submission 2 corresponds to one independently trained model, using self-ensembling and using 200k words for source vocabulary and 50k words for target vocabulary.

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

Subtask	Ja → En			En → Ja		Ja → Zh		Zh → Ja		
Model	EBMT	NMT		EBMT	NMT	EBMT	NMT	EBMT	NMT	
Human Evaluation		1	2		1		1		1	2
BLEU	21.22	24.71	26.22	31.03	36.19	30.27	31.98	36.63	46.04	44.29
RIBES	70.57	75.08	75.66	77.12	81.98	81.31	83.76	82.03	87.65	86.94
AM-FM	59.52	56.27	55.85	74.75	73.87	76.42	76.33	76.71	78.59	78.44
Pairwise	-	47.00	44.25	-	55.25	30.75	58.75	-	63.75	56.00
Rank	-	3/9	4/9	-	1/10	3/5	1/5	-	1/9	2/9
JPO adequacy	-	3.89	-	-	4.02	-	3.88	-	3.94	-
Rank	-	1/3	-	-	1/4	-	1/3	-	1/3	-

Table 1: Official automatic and human evaluation results of our EBMT and NMT systems for the ASPEC subtasks.

Source	本フローセンサーの型式と基本構成，規格を 図示，紹介。
Reference	Shown here are type and basic configuration and standards of this flow **with some diagrams**.
EBMT	This flow sensor type and the basic composition, standard **is illustrated**, and introduced.
NMT	This paper introduces the type, basic configuration, and standards of this flow sensor.

Table 2: A Japanese-to-English translation example by our EBMT and NMT systems.

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.

In view of the pairwise evaluation, our system achieved the best translation quality for all the subtasks except for Ja → En. The difference of the pairwise scores between the best system and our system for Ja → En is 1.25, which is *not* statistically significant. For all the other subtasks, the differences between our system and others are statistically significant ($p < 0.01$).

As for the JPO adequacy evaluation, our system achieved the best translation quality for all the subtasks. The differences of the score between our system and the 2nd graded systems are 0.058, 0.305, 0.245 and 0.300 for Ja → En, En → Ja, Ja → Zh and Zh → Ja respectively.

4.3 Error Analysis

We analyzed the translation results of both our EBMT and NMT systems. We found that the biggest problem of EBMT is word order errors, which affects the fluency and meaning of the translations. This is due to the reason that the word order of the translations in EBMT depends on the parse trees of the input sentences, but the parsing accuracy is not perfect especially for Chinese. NMT tends to produce fluent translations, however it lacks of adequacy sometimes.

The most common problem of NMT is that it could produce under or over translated translations, due to the lack of a way for the attention mechanism to memorize the source words that have been translated during decoding. We plan to address this problem with the coverage model proposed in (Tu et al., 2016). The UNK words are also a big problem. Although we deal with them using the UNK replacement method (Luong et al., 2015), it could simply fail because of errors for finding the corresponding source words using attention.

We show a Japanese-to-English translation example by our EBMT and NMT systems in Table 2 to illustrate some of the above problems. The translation produced by the EBMT system has a word order problem that changes the meaning, making "the basic composition, standard" independent from "this flow sensor." It also has an agreement violation problem between the argument and the predicate that "is illustrated" should be "are illustrated". The translation produced by the NMT system is more fluent, but it does not translate the source word "図示 (be illustrated)". In addition, it adds the additional information "this paper" to the translation.

It is interesting to note that the performance gap between NMT and EBMT is largest for Zh → Ja, when Chinese is also the language most difficult to parse. The gap is smaller when Japanese, the easiest language to parse, is the source language. We can infer that the EBMT system is somehow quite handicapped by source sentence parsing issues, as we had noted in our previous results.

5 Conclusion

We have described our experiments for WAT 2016 using both an EBMT system and a NMT system. We could obtain the best or close-to-best results in each translation task. The NMT approach proved to be quite successful, which is in line with other recent MT evaluation results (Bojar et al., 2016). In the future, we will probably continue to explore the NMT approach, if possible merging in some elements of our EBMT system. We could then hope to solve the weaknesses of the two systems that we identified in our error analysis.

References

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

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation (wmt16). In *Proceedings of WMT 2016*.

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*.

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

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

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*.

Thang Luong, Kyunghyun Cho, and Christopher D. Manning. 2016. Neural machine translation (ACL tutorial). *https://sites.google.com/site/acl16nmt/*.

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)*, Osaka, Japan, October.

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 (LREC2016)*, Portoroz, Slovenia, 5.

John Richardson, Raj Dabre, Chenhui Chu, Fabien Cromières, Toshiaki Nakazawa, and Sadao Kurohashi. 2015. KyotoEBMT System Description for the 2nd Workshop on Asian Translation. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 54–60, Kyoto, Japan, October.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the first Conference on Machine Translation (WMT2016)*.

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, December. 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)*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany, August. Association for Computational Linguistics.

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

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.