# A Simple and Strong Baseline: NAIST-NICT Neural Machine Translation System for WAT2017 English-Japanese Translation Task

**Yusuke Oda**[†‡] **Katsuhito Sudoh**[†] **Satoshi Nakamura**[†]
**Masao Utiyama**[‡] **Eiichiro Sumita**[‡]
† Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara 630-0192, Japan
‡ National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Kyoto 619-0289, Japan

## Abstract

This paper describes the details about the NAIST-NICT machine translation system for WAT2017 English-Japanese Scientific Paper Translation Task. The system consists of a language-independent tokenizer and an attentional encoder-decoder style neural machine translation model. According to the official results, our system achieves higher translation accuracy than any systems submitted previous campaigns despite simple model architecture.

## 1 Introduction

Neural machine translation (NMT) methods became one of the main-stream techniques in current machine translation studies. Previous WAT campaign showed that NMT methods can achieve higher translation accuracy in spite of simple model configurations (Nakazawa et al., 2016a). In this year, we chose the NMT architecture as our translation systems submitted for WAT2017 English-Japanese Scientific Paper Translation Task (Nakazawa et al., 2017). The main translation model is constructed by an encoder-decoder model (Sutskever et al., 2014) enforced by an attention mechanism (Bahdanau et al., 2014; Luong et al., 2015). This paper describes the details of our system, including whole model architecture, training criteria, decoding strategy, and data preparation. Results show that our system achieves higher translation accuracy than any systems submitted in previous WAT campaigns.

## 2 Architecture of the System

### 2.1 Model formulation

Our translation model is built by an attentional encoder-decoder network implemented in NMTKit [1]. Overall model structure is based on the combination of Bahdanau et al. (2014) and Luong et al. (2015). This translation model represents a conditional probability $\Pr(\boldsymbol{e}|\boldsymbol{f})$, where $\boldsymbol{e} := [e_1, e_2, \cdots, e_E]$ and $\boldsymbol{f} := [f_1, f_2, \cdots, f_F]$ are sequences of target/source words, and $E$, $F$ are the numbers of the target/source words. Encoder-decoder style NMT models behaves a sequential word generators, which yields target words one-by-one along the time step. Formally, the whole translation model is separated into the product of token-wise conditional probabilities:

$$\Pr(\boldsymbol{e}|\boldsymbol{f}) = \prod_{t=1}^{E} \Pr(e_t|\boldsymbol{e}_{<t}, \boldsymbol{f}), \tag{1}$$

where $\boldsymbol{e}_{<t} := [e_1, \cdots, e_t]$ is the history of generated target words. In each time step $t$, the condition $\langle \boldsymbol{e}_{<t}, \boldsymbol{f} \rangle$ is represented as a *condition* vector $\boldsymbol{\eta}_t$ and each conditional probability in Equation (1) is calculated by the softmax function as follows:

$$\begin{aligned} \Pr(e_t|\boldsymbol{e}_{<t}, \boldsymbol{f}) &:= \Pr(e_t|\boldsymbol{\eta}_t) && (2) \\ &:= \mathrm{softmax}(\mathbf{W}_e \boldsymbol{\eta}_t + \boldsymbol{b}_e). && (3) \end{aligned}$$

where $\mathbf{W}_e$ and $\boldsymbol{b}_e$ are the parameters to be trained: weight matrix and bias vector respectively. We calculate the condition vector $\boldsymbol{\eta}_t$ using three sub-models: *encoder*, *decoder* and *attention* described in the following sections.

#### 2.1.1 Encoder

The *encoder* converts given source sequence $\boldsymbol{f}$ to a set of vectors $\boldsymbol{R} := [\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_F]$. Each vector at the position $i$ is calculated as follows:

$$\begin{aligned} \boldsymbol{r}_i &:= \mathrm{concat}(\overrightarrow{\boldsymbol{r}}_i, \overleftarrow{\boldsymbol{r}}_i), && (4) \\ \overrightarrow{\boldsymbol{r}}_i &:= \mathrm{RNN}(\mathrm{emb}(f_i), \overrightarrow{\boldsymbol{r}}_{i-1}) && (5) \\ \overleftarrow{\boldsymbol{r}}_i &:= \mathrm{RNN}(\mathrm{emb}(f_i), \overleftarrow{\boldsymbol{r}}_{i+1}), && (6) \end{aligned}$$

[1] https://github.com/odashi/nmtkit



*Proceedings of the 4th Workshop on Asian Translation*, pages 135–139,
Taipei, Taiwan, November 27, 2017. 

where $\mathrm{concat}(\cdots)$ represents the concatenation of given vectors, $\mathrm{emb}(w)$ represents the lookup of embedding vectors corresponding to the token $w$, and $\mathrm{RNN}(\cdot,\cdot)$ represents an independent recurrent unit. We use the long short-term memory units (Gers et al., 2000) with input/forget/output gates to all recurrent units. We also introduce the $n$-*stacked encoder* to represent richer source information as follows:

$$\boldsymbol{r}_i := \mathrm{concat}(\overrightarrow{\boldsymbol{r}}_i^{(n)}, \overleftarrow{\boldsymbol{r}}_i^{(n)}), \quad (7)$$

$$\overrightarrow{\boldsymbol{r}}_i^{(n)} := \mathrm{RNN}(\overrightarrow{\boldsymbol{r}}_i^{(n-1)}, \overrightarrow{\boldsymbol{r}}_{i-1}^{(n)}), \quad (8)$$

$$\overleftarrow{\boldsymbol{r}}_i^{(n)} := \mathrm{RNN}(\overleftarrow{\boldsymbol{r}}_i^{(n-1)}, \overleftarrow{\boldsymbol{r}}_{i+1}^{(n)}), \quad (9)$$

$$\overrightarrow{\boldsymbol{r}}_i^{(0)} := \mathrm{emb}(f_i), \quad (10)$$

$$\overleftarrow{\boldsymbol{r}}_i^{(0)} := \mathrm{emb}(f_i). \quad (11)$$

We set all initial recurrent states $\overrightarrow{\boldsymbol{r}}_0^{(n)}$ and $\overleftarrow{\boldsymbol{r}}_{F+1}^{(n)}$ to $\mathbf{0}$.

### 2.1.2 Decoder and attention

The *decoder* calculates the condition vector $\boldsymbol{\eta}_t$ as follows:

$$\boldsymbol{\eta}_t := \tanh(\mathbf{W}_\eta \mathrm{concat}(\boldsymbol{c}_t, \boldsymbol{h}_t) + \boldsymbol{b}_\eta), \quad (12)$$

where $\mathbf{W}_\eta$ and $\boldsymbol{b}_\eta$ are the parameters. $\boldsymbol{h}_t$ is the current decoder's state calculated by a unidirectional recurrent unit:

$$\boldsymbol{h}_t := \mathrm{RNN}(\mathrm{concat}(\mathrm{emb}(e_{t-1}), \boldsymbol{\eta}_{t-1}), \boldsymbol{h}_{t-1})), \quad (13)$$

and we extend this formulation to the $n$-*stacked* version with the notation $\boldsymbol{h}_t^{(n)}$ by the similar modification to that of the encoder. In this calculation, we also introduce the previous condition vector $\boldsymbol{\eta}_{t-1}$ as an additional input of the recurrent unit. This is called as the *input feeding* (Luong et al., 2015), allows to propagate previous decision of the decoder with keeping differentiability of the network. $\boldsymbol{c}_t$ is the *context* vector calculated from $\boldsymbol{R}$ using an *attention* mechanism:

$$\boldsymbol{c}_t := \mathbf{R}\boldsymbol{a}_t, \quad (14)$$

where $\mathbf{R}$ is a matrix created by substituting all vectors $\boldsymbol{r}_i$ to the $i$-th columns. $\boldsymbol{a}_t$ represents the weight of each vector $\boldsymbol{r}_i$ at the time $t$, which is calculated by an arbitrary *score* function $\boldsymbol{\alpha}$:

$$\boldsymbol{a}_t := \mathrm{softmax}(\boldsymbol{\alpha}(\mathbf{R}, \boldsymbol{h}_t)). \quad (15)$$

We follow the multi-layer perceptron based score function proposed by Bahdanau et al. (2014):

$$\alpha_i(\mathbf{R}, \boldsymbol{h}_t) := \boldsymbol{v}_\alpha^\top \tanh(\mathbf{W}_\alpha \mathrm{concat}(\boldsymbol{r}_i, \boldsymbol{h}_t)), \quad (16)$$

where $\alpha_i$ represents the $i$-th element of $\boldsymbol{\alpha}$, $\boldsymbol{v}_\alpha$ and $\mathbf{W}_\alpha$ are the parameters.

For the initial values $\boldsymbol{h}_0^{(n)}$, we use the *dense bridge* connection from the encoder units as follows:

$$\boldsymbol{h}_0^{(n)} := \tanh(\boldsymbol{s}_{\boldsymbol{h},0}^{(n)}), \quad (17)$$

$$\boldsymbol{s}_{\boldsymbol{h},0}^{(n)} := \mathbf{U}^{(n)} \boldsymbol{s}_{\overrightarrow{\boldsymbol{r}},F}^{(n)} + \mathbf{V}^{(n)} \boldsymbol{s}_{\overleftarrow{\boldsymbol{r}},1}^{(n)} + \boldsymbol{b}_{h0}^{(n)}, \quad (18)$$

where $\mathbf{U}^{(n)}$, $\mathbf{V}^{(n)}$ and $\boldsymbol{b}_{h0}^{(n)}$ are the parameters, and $\boldsymbol{s}_{\boldsymbol{x},t}^{(n)}$ denotes the internal states of the LSTMs corresponding to the outputs $\boldsymbol{x}_t^{(n)}$.

### 2.1.3 Hyper-parameters

The translation model described in previous sections has several hyper-parameters: number of units in source/target embeddings $\mathrm{emb}(\cdot)$, RNN states $\overrightarrow{\boldsymbol{r}}$, $\overleftarrow{\boldsymbol{r}}$ and $\boldsymbol{h}$, the hidden layer in the attention network, and the condition vector $\boldsymbol{\eta}$. We constrained all these numbers same to prevent increasing combination of hyper-parameters. In addition, each recurrent unit also has the depth of stack as an additional hyper-parameter. We also constrained these numbers same due to the Equations (17) and (18). Eventually, we varied the number of units from 256 to 1024 and depth of the recurrent stack from 1 to 4 to construct models that have a different power of model expressiveness and finally selected 512 as the former and 2 as the latter hyper-parameter respectively.

## 2.2 Model Training

To find an optimal parameters in the model described in the previous sections, we minimize the cross-entropy loss function:

$$\mathcal{L}(\theta) := -\sum_m \sum_t \log \mathcal{P}_{m,t}(\theta), \quad (19)$$

$$\mathcal{P}_{m,t}(\theta) := \Pr(e_t = e_t^{(m)} | \boldsymbol{e}_{<t} = \boldsymbol{e}_{<t}^{(m)}, \boldsymbol{f} = \boldsymbol{f}^{(m)}; \theta), \quad (20)$$

where $\boldsymbol{f}^{(m)}$ and $\boldsymbol{e}^{(m)}$ represents the $m$-th parallel corpora, and $\theta$ represents the set of all parameters in a translation model.

To achieve this optimization problem, we use the Adam optimizer (Kingma and Ba, 2014) for all training settings.

**Hyper-parameters** Since Adam has several hyper-parameters that may directly affect convergence speed and model quality, we tried to train

our models with various combination of hyper-parameters, and finally chose as follows: $\alpha = 0.0001$ (default/10), $\beta_1 = 0.9$ (default), $\beta_2 = 0.999$ (default), and $\epsilon = 10^{-8}$ (default).

### 2.3 Tokenization

Tokenization is one of the worrisome problems of machine translation systems. Since there are no linguistically unified criteria about separating words from original sentences, we often have to choose tokenization methods carefully according to selected languages. In contrast, we use the SentencePiece tokenizer[2] instead of some language-dependent tokenization methods. This tokenizer basically requires only an *untokenized* training corpora to acquire actual tokenization strategies and frees us from selecting actual tokenizers. In our system, all raw sentences are tokenized using the SentencePiece tokenizer trained by the corpus used to train translation models, and resulting tokens generated by the translation models are detokenized by simply removing boundary markers in the concatenated strings. Through the whole process of our translation systems, we never use any other pre/post-processing methods.

**Hyper-parameters** SentencePiece requires the vocabulary size $V$ as the hyper-parameter. We tried to use $V = 4096, 8192, 16384, 32768$, and finally chose $V = 16384$ for each language.

### 2.4 Decoding

In decoding actual target sentences, we performed greedy $n$-best beam search method with the *word penalty* heuristic, which simply multiplies the constant $\exp(WP)$ to all word probabilities in each time. The word penalty introduces an exponential distribution as a prior knowledge about the length of the target sentence. If we set the penalty factor $WP > 0$, then the system penalizes shorter sentences, and tends to generate longer sentences. Note that if the beam width is 1, there is no effect from word penalty, because the translation system can generate only 1-best results.

**Hyper-parameters** In our decoding strategy, We have 2 hyper-parameters: beam width $BW$ and word penalty factor $WP$. We varied $BW$ from 1 to 128, and $WP$ from 0 to 1.5, and finally chose $BW = 16$ and $WP = 0.75$.

[2]https://github.com/google/sentencepiece

Table 1: Official evaluation results of our systems.

| System | BLEU | Place (All / Single) |
|---|---|---|
| One-best | 36.47 | 11 / 6 |
| Adjusted | **38.25** | 8 / 3 |
| (last-year) | 36.19 | — / — |
| **System** | **RIBES** | **Place (All / Single)** |
| One-best | 0.821989 | 13 / 6 |
| Adjusted | **0.834492** | 5 / 2 |
| (last-year) | 0.819836 | — / — |
| **System** | **AM-FM** | **Place (All / Single)** |
| One-best | 0.763310 | 5 / 4 |
| Adjusted | **0.770480** | 1 / 1 |
| (last-year) | 0.758740 | — / — |
| **System** | **Human** | **Place (All / Single)** |
| One-best | 63.500 | 8 / 3 |
| Adjusted | **70.000** | 4 / 1 |
| (last-year) | — | — / — |

### 2.5 Model Ensembling

Although the model ensembling techniques improve the translation accuracy, they also impose a great deal of computation back-ends (e.g., if a single model requires a full resource of one GPU, the $N$-ensemble system basically occupies $N$ GPUs while executing it) and this behavior is typically not fit to the most situations of real production systems. Because of this issue, we did not introduce any model ensembling techniques while decoding test inputs.

## 3 Results

We trained all translation systems varied by model/training/tokenization hyper-parameters described in previous sections, and performed a grid search to find an optimal set of hyper-parameters for this task. For the training data, we used top 2M sentences in ASPEC corpus (Nakazawa et al., 2016b) provided by the organizer. We chose the optimal model that achieves the best BLEU (Papineni et al., 2002) score over the *dev* corpus. For the optimal model, we also performed a grid search about decoding-time hyper-parameters. All the optimal hyper-parameters described in previous sections are found as the results of these searches.

We submitted two results generated from the same optimal model: *one-best* results, i.e., the results with fixing $BW = 1$, and *adjusted* results, i.e., the results with optimal $BW$ and $WP$ de-

Table 2: JPO adequacy results.

| System | Ensemble | Scores | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| (this-year) | 8 models | 0.25 | 1.75 | 8.25 | 36.50 | 53.25 |
| Adjusted | Single | 0.25 | 1.75 | 17.50 | 37.75 | 42.75 |
| (last-year) | 3 models | 2.00 | 2.75 | 19.25 | 43.50 | 32.50 |

scribed in Section 2.4.

Table 1 shows the official evaluation scores of our systems, including BLEU, RIBES (Isozaki et al., 2010), AM-FM (Banchs and Li, 2011), and the human evaluation. The rows labeled *last-year* shows the best system in all previous WAT campaigns. We can see that our *one-best* system already achieves higher translation accuracy in all automatic evaluation metrics than *last-year* systems. In addition, *adjusted* system achieves further better scores than *one-best*, which means applying better decoding strategy can improve translation accuracy even using the same model.

Table 1 also shows the place of our systems in this year. Because official results do not separate scores of single (no-ensemble) models and ensemble models, we also calculated the place of our systems out of only single models for fair comparison. In AM-FM and human evaluation, we can see that our *adjusted* model marks the 1st place of this year's campaign.

Table 2 shows the results of the JPO pairwise adequacy evaluation provided by the organizer. We also showed the *this-year* system, the 1st place system (by NTT team) of the same task in this year's campaign, as well as the *Adjusted* and the *last-year* systems. By comparing with *Adjusted* and *last-year*, we can see that our system clearly increases the number of the highest (5) score and reduces all other (1 to 4) scores. In particular, our system reduces the number of lower-range (1 or 2) scores by the same level of the *this-year* system although we did not use model ensembling. However, there is still room for improvement about higher-range (3 to 5) scores.

## 4 Conclusion

This paper described the details of the NAIST-NICT neural machine translation systems submitted to WAT2017 English-Japanese Scientific Paper Translation Task. Although the model structure is not new, our model achieved higher translation accuracy compared with past systems in this language pair. In addition, we also tried to use SentencePiece, an unsupervised tokenizer to avoid complicated tokenization problems, and also confirmed that the resulting translation systems can perform with no accuracy reduction.

## Acknowledgement

Part of this work was supported by JSPS KAKENHI Grant Numbers JP17H06101 and JP16H05873.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Rafael E. Banchs and Haizhou Li. 2011. Am-fm: A semantic framework for translation quality assessment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 153–158, Portland, Oregon, USA. Association for Computational Linguistics.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471.

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. Association for Computational Linguistics.

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

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, Lisbon, Portugal. Association for Computational Linguistics.

Toshiaki Nakazawa, Chenchen Ding, Hideya MINO, Isao Goto, Graham Neubig, and Sadao Kurohashi. 2016a. Overview of the 3rd workshop on asian translation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 1–46, Osaka, Japan. The COLING 2016 Organizing Committee.

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, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016b. Aspec: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208, Portoro, Slovenia. European Language Resources Association (ELRA).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.