The AMU-UEdin Submission to the WMT 2017 Shared Task on Automatic Post-Editing

Marcin Junczys-Dowmunt
Information Systems Laboratory
Adam Mickiewicz University in Poznań
junczys@amu.edu.pl

Roman Grundkiewicz
School of Informatics
University of Edinburgh
rgrundki@exseed.ed.ac.uk

Abstract

This work describes the AMU-UEdin submission to the WMT 2017 shared task on Automatic Post-Editing. We explore multiple neural architectures adapted for the task of automatic post-editing of machine translation output. We focus on neural end-to-end models that combine both inputs mt and src in a single neural architecture, modeling $\{mt, src\} \rightarrow pe$ directly. Apart from that, we investigate the influence of hard-attention models which seem to be well-suited for monolingual tasks, as well as combinations of both ideas.

1 Introduction

During the WMT 2016 APE two systems relied on neural models, the CUNI system (Libovický et al., 2016) and the shared task winner, the system submitted by the Adam Mickiewicz University (AMU) team (Junczys-Dowmunt and Grundkiewicz, 2016). This submission explored the application of neural translation models to the APE problem and achieved good results by treating different models as components in a log-linear model, allowing for multiple inputs (the source src and the translated sentence mt) that were decoded to the same target language (post-edited translation pe). Two systems were considered, one using src as the input ($src \rightarrow pe$) and another using mt as the input ($mt \rightarrow pe$). A simple string-matching penalty integrated within the log-linear model was used to control for higher faithfulness with regard to the raw MT output. The penalty fires if the APE system proposes a word in its output that has not been seen in mt. The influence of the components on the final result was tuned with Minimum Error Rate Training (Och, 2003) with regard to the task metric TER.

With neural encoder-decoder models, and multi-source models in particular, the combination of mt and src can be now achieved in more natural ways than for previously popular phrase-based statistical machine translation (PB-SMT) systems. Despite this, results for multi-source or double-source models in APE scenarios are incomplete or unsatisfying in terms of performance.

In this work, we explore a number of single-source and double-source neural architectures which we believe to be better fits to the APE task than vanilla encoder-decoder models with soft attention. We focus on neural end-to-end models that combine both inputs mt and src in a single neural architecture, modeling $\{mt, src\} \rightarrow pe$ directly. Apart from that, we investigate the influence of hard-attention models which seem to be well-suited for monolingual tasks. Finally, we create combinations of both architectures.

Following (Junczys-Dowmunt and Grundkiewicz, 2016), we also attempt to generate more artificial data for the task. Instead of relying on filtering towards specific error rates, we generate text with fitting error rates from the start which allows us to retain more data.

2 Encoder-Decoder Models with APE-specific Attention Models

2.1 Standard Attentional Encoder-Decoder

The attentional encoder-decoder model in Marian[1] is a re-implementation of the NMT model in Nematus (Sennrich et al., 2017). The model differs from the standard model introduced by Bahdanau et al. (2014) by several aspects, the most important being the conditional GRU with attention. The summary provided in this section is based on the description in Sennrich et al. (2017). More details on the specific architectures in this shared

[1] https://github.com/marian-nmt/marian

task submission are given in Junczys-Dowmunt and Grundkiewicz (2017).

Given the raw MT output sequence $(x_1, \dots, x_{T_x})$ of length T_x and its manually post-edited equivalent $(y_1, \dots, y_{T_y})$ of length T_y, we construct the encoder-decoder model using the following formulations.

Encoder context A single forward encoder state $\overrightarrow{\mathbf{h}}_i$ is calculated as:

$$\overrightarrow{\mathbf{h}}_i = \text{GRU}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{F}[x_i])$$

where $\mathbf{F}$ is the encoder embeddings matrix. The GRU RNN cell (Cho et al., 2014) is defined as:

$$\begin{aligned} \text{GRU}\left(\mathbf{s}, \mathbf{x}\right) =& (1 - \mathbf{z}) \odot \underline{\mathbf{s}} + \mathbf{z} \odot \mathbf{s}, \qquad (1) \\ \underline{\mathbf{s}} =& \tanh\left(\mathbf{W}\mathbf{x} + \mathbf{r} \odot \mathbf{U}\mathbf{s}\right), \\ \mathbf{r} =& \sigma\left(\mathbf{W}_r\mathbf{x} + \mathbf{U}_r\mathbf{s}\right), \\ \mathbf{z} =& \sigma\left(\mathbf{W}_z\mathbf{x} + \mathbf{U}_z\mathbf{s}\right), \end{aligned}$$

where $\mathbf{x}$ is the cell input, $\mathbf{s}$ is the previous recurrent state, $\mathbf{W}$, $\mathbf{U}$, $\mathbf{W}_r$, $\mathbf{U}_r$, $\mathbf{W}_z$, $\mathbf{U}_z$ are trained model parameters[2]; σ is the logistic sigmoid activation function. The backward encoder state is calculated analogously over a reversed input sequence with its own set of trained parameters.

Let $\mathbf{h}_i$ be the annotation of the source symbol at position i, obtained by concatenating the forward and backward encoder RNN hidden states, $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$, the set of encoder states $\mathrm{C} = \{\mathbf{h}_1, \dots, \mathbf{h}_{T_x}\}$ then forms the encoder context.

Decoder initialization The decoder is initialized with start state s_0, computed as the average over all encoder states:

$$\mathbf{s}_0 = \tanh\left(\mathbf{W}_{init} \frac{\sum_{i=1}^{T_x} \mathbf{h}_i}{T_x}\right)$$

Conditional GRU with attention We follow the Nematus implementation of the conditional GRU with attention, cGRU$_{\text{att}}$:

$$\mathbf{s}_j = \text{cGRU}_{\text{att}}\left(\mathbf{s}_{j-1}, \mathbf{E}[y_{j-1}], \mathrm{C}\right) \qquad (2)$$

where $\mathbf{s}_j$ is the newly computed hidden state, $\mathbf{s}_{j-1}$ is the previous hidden state, C the source context and $\mathbf{E}[y_{j-1}]$ is the embedding of the previously decoded symbol y_{i-1}.

The conditional GRU cell with attention, cGRU$_{\text{att}}$, has a complex internal structure, consisting of three parts: two GRU layers and an intermediate attention mechanism ATT.

[2]Biases have been omitted.

Layer GRU$_1$ generates an intermediate representation $\mathbf{s}'_j$ from the previous hidden state $\mathbf{s}_{j-1}$ and the embedding of the previous decoded symbol $\mathbf{E}[y_{j-1}]$:

$$\mathbf{s}'_j = \text{GRU}_1\left(\mathbf{s}_{j-1}, \mathbf{E}[y_{j-1}]\right).$$

The attention mechanism, ATT, inputs the entire context set C along with intermediate hidden state $\mathbf{s}'_j$ in order to compute the context vector $\mathbf{c}_j$ as follows:

$$\begin{aligned} \mathbf{c}_j =& \text{ATT}\left(\mathrm{C}, \mathbf{s}'_j\right) = \sum_{i}^{T_x} \alpha_{ij}\mathbf{h}_i, \\ \alpha_{ij} =& \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{kj})}, \\ e_{ij} =& \mathbf{v}_a^\intercal \tanh\left(\mathbf{U}_a\mathbf{s}'_j + \mathbf{W}_a\mathbf{h}_i\right), \end{aligned}$$

where α_{ij} is the normalized alignment weight between source symbol at position i and target symbol at position j and $\mathbf{v}_a, \mathbf{U}_a, \mathbf{W}_a$ are trained model parameters.

Layer GRU$_2$ generates $\mathbf{s}_j$, the hidden state of the cGRU$_{\text{att}}$, from the intermediate representation $\mathbf{s}'_j$ and context vector $\mathbf{c}_j$:

$$\mathbf{s}_j = \text{GRU}_2\left(\mathbf{s}'_j, \mathbf{c}_j\right).$$

Deep output Finally, given $\mathbf{s}_j$, y_{j-1}, and $\mathbf{c}_j$, the output probability $p(y_j|\mathbf{s}_j, y_{j-1}, \mathbf{c}_j)$ is computed by a softmax activation as follows:

$$\begin{aligned} p(y_j|\mathbf{s}_j, y_{j-1}, \mathbf{c}_j) &= \text{softmax}\left(\mathbf{t}_j\mathbf{W}_o\right) \\ \mathbf{t}_j &= \tanh\left(\mathbf{s}_j\mathbf{W}_{t_1} + \mathbf{E}[y_{j-1}]\mathbf{W}_{t_2} + \mathbf{c}_j\mathbf{W}_{t_3}\right) \end{aligned}$$

$\mathbf{W}_{t_1}, \mathbf{W}_{t_2}, \mathbf{W}_{t_3}, \mathbf{W}_o$ are the trained model parameters.

This rather standard encoder-decoder model with attention is our baseline and denoted as ENCDEC-ATT.

The following models reuse most parts of the architecture described above wherever possible, most differences occur in the decoder RNN cell and the attention mechanism. The encoders are identical, so are the deep output layers.

2.2 Hard Monotonic Attention

Aharoni and Goldberg (2016) introduce a simple model for monolingual morphological re-inflection with hard monotonic attention. This model looks at one encoder state at a time, starting with the left-most encoder state and progressing to the right until all encoder states have been processed.

The target word vocabulary V_y is extended with a special step symbol ($V'_y = V_y \cup \{\langle\textsc{step}\rangle\}$) and whenever $\langle\textsc{step}\rangle$ is predicted as the output symbol, the hard attention is moved to the next encoder state. Formally, the hard attention mechanism is represented as a precomputed monotonic sequence $(a_1, \ldots, a_{T_y})$ which can be inferred from the target sequence $(y_1, \ldots, y_{T_y})$ (containing original target symbols and T_x step symbols) as follows:

$$a_1 = 1$$
$$a_j = \begin{cases} a_{j-1} + 1 & \text{if } y_{j-1} = \langle\textsc{step}\rangle \\ a_{j-1} & \text{otherwise.} \end{cases}$$

For a given context $\mathrm{C} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{T_x}\}$, the attended context vector at time step j is simply h_{a_j}.

Following the description by Aharoni and Goldberg (2016) for their LSTM-based model, we now adapt the previously described encoder-decoder model to incorporate hard attention. The encoder as well as the output layer of the previous model remain unchanged. Given the sequence of attention indices $(a_1, \ldots, a_{T_y})$, the conditional GRU cell (Eq. 2) used for hidden state updates of the decoder is replaced with a simple GRU cell (Eq. 1) (thus removing the soft-attention mechanism):

$$\mathbf{s}_j = \mathrm{GRU}\left(\mathbf{s}_{j-1}, \left[\mathbf{E}[y_{j-1}]; \mathbf{h}_{a_j}\right]\right) \qquad (3)$$

where the cell input is now a concatenation of the embedding of the previous target symbol $E[y_{j-1}]$ and the currently attended encoder state $\mathbf{h}_{a_j}$. This model is labeled ENCDEC-HARD.

We find this architecture compelling for monolingual tasks that might require higher faithfulness with regard to the input. With hard monotonic attention, the translation algorithm can enforce certain constraints:

1. The end-of-sentence symbol can only be generated if the hard attention mechanism has reached the end of the input sequence, enforcing full coverage;

2. The $\langle\textsc{step}\rangle$ symbol cannot be generated once the end-of-sentence position in the source has been reached. It is however still possible to generate content tokens.

Obviously, this model requires a target sequence with correctly inserted $\langle\textsc{step}\rangle$ symbols. For the described APE task, using the Longest Common Subsequence algorithm (Hirschberg, 1977), we first generate a sequence of match, delete and insert operations which transform the raw MT output $(x_1, \cdots x_{T_x})$ into the corrected post-edited sequence $(y_1, \cdots y_{T_y})$[3]. Next, we map these operations to the final sequence of steps and target tokens according to the following rules:

- For each matched pair of tokens x, y we produce symbols: $\langle\textsc{step}\rangle$ y;
- For each inserted target token y, we produce the same token y;
- For each deleted source token x we produce $\langle\textsc{step}\rangle$;
- Since at initialization of the model $a_1 = 1$, i.e. the first encoder state is already attended to, we discard the first symbol in the new sequence if it is a $\langle\textsc{step}\rangle$ symbol.

2.3 Hard and Soft Attention

While the hard attention model can be used to enforce faithfulness to the original input, we would also like the model to be able to look at information anywhere in the source sequence which is a property of the soft attention model.

By re-introducing the conditional GRU cell with soft attention into the ENCDEC-HARD model while also inputting the hard-attended encoder state h_{a_j}, we can try to take advantage of both attention mechanisms. Combining Eq. 2 and Eq. 3, we get:

$$\mathbf{s}_j = \mathrm{cGRU}_{\mathrm{att}}\left(\mathbf{s}_{j-1}, \left[\mathbf{E}[y_{j-1}]; \mathbf{h}_{a_j}\right], \mathrm{C}\right). \qquad (4)$$

The rest of the model is unchanged; the translation process is the same as before and we use the same target step/token sequence for training. This model is called ENCDEC-HARD-ATT.

2.4 Soft Double-Attention

Neural multi-source models (Zoph and Knight, 2016) seem to be natural fit for the APE task, as raw MT output and original source language input are available. Although application to the APE problem have been reported (Libovický and Helcl, 2017), state-of-the-art results seem to be missing.

In this section we give details about our double-source model implementation. We rename the existing encoder C to C^{mt} to signal that the first encoder consumes the raw MT output and introduce

[3] Similar to GNU wdiff.

a structurally identical second encoder $C^{src} = \{\mathbf{h}_1^{src}, \ldots, \mathbf{h}_{T_{src}}^{src}\}$ over the source language. To compute the decoder start state s_0 for the multi-encoder model we concatenate the averaged encoder contexts before mapping them into the decoder state space:

$$\mathbf{s}_0 = \tanh\left(\mathbf{W}_{init}\left[\frac{\sum_{i=1}^{T_{mt}}\mathbf{h}_i^{mt}}{T_{mt}}; \frac{\sum_{i=1}^{T_{src}}\mathbf{h}_i^{src}}{T_{src}}\right]\right).$$

In the decoder, we replace the conditional GRU with attention, with a doubly-attentive cGRU cell (Calixto et al., 2017) over contexts C^{mt} and C^{src}:

$$\mathbf{s}_j = \text{cGRU}_{\text{2-att}}\left(\mathbf{s}_{j-1}, \mathbf{E}[y_{j-1}], \mathrm{C}^{mt}, \mathrm{C}^{src}\right) \quad (5)$$

The procedure is similar to the original cGRU, differing only in that in order to compute the context vector $\mathbf{c}_j$, we first calculate contexts vectors $\mathbf{c}_j^{mt}$ and $\mathbf{c}_j^{src}$ for each context and then concatenate the results:

$$\begin{aligned}
\mathbf{s}'_j &= \text{GRU}_1\left(\mathbf{s}_{j-1}, \mathbf{E}[y_{j-1}]\right), \\
\mathbf{c}_j^{mt} &= \text{ATT}\left(\mathrm{C}^{mt}, \mathbf{s}'_j\right) = \sum_i^{T_{mt}} \alpha_{ij}\mathbf{h}_i^{mt}, \\
\mathbf{c}_j^{src} &= \text{ATT}\left(\mathrm{C}^{src}, \mathbf{s}'_j\right) = \sum_i^{T_{src}} \alpha_{ij}\mathbf{h}_i^{src}, \\
\mathbf{c}_j &= \left[\mathbf{c}_j^{mt}; \mathbf{c}_j^{src}\right], \\
\mathbf{s}_j &= \text{GRU}_2\left(\mathbf{s}'_j, \mathbf{c}_j\right).
\end{aligned}$$

This could be easily extended to an arbitrary number of encoders with different architectures. During training this model is fed with a tri-parallel corpus, during translation both input sequences are processed simultaneously to produce the corrected output. This model is denoted as ENCDEC-DOUBLE-ATT.

2.5 Hard Attention with Soft Double-Attention

Analogously to the procedure described in section 2.3, we can extend the doubly-attentive cGRU to take the hard-attended encoder context as additional input:

$$\mathbf{s}_j = \text{cGRU}_{\text{2-att}}\left(\mathbf{s}_{j-1}, \left[\mathbf{E}[y_{j-1}]; \mathbf{h}_{a_j}^{mt}\right], \mathrm{C}^{mt}, \mathrm{C}^{src}\right)$$

In this formulation, only the first encoder context C^{mt} is attended to by the hard monotonic attention mechanism. The target training data consists of the step/token sequences used for all previous hard-attention models. We call this model ENCDEC-HARD+DOUBLE-ATT.

Data set	Sentences	TER
training set 2016	12,000	26.22
training set 2017	11,000	–
development set 2016	1,000	24.81
test set 2016	2,000	–
artificial-large 2016	4,335,715	36.63
artificial-small 2016	531,839	25.28
artificial 2017	15,158,354	27.45

Table 1: Statistics for artificial data sets in comparison to official training and development data, adapted from Junczys-Dowmunt and Grundkiewicz (2016).

3 Artifical Data

We also attempt to generate more artificial data for the task. Instead of relying on filtering towards specific error rates, we generate text with fitting error rates from the start which allows us to retain more data. To obtain the monolingual source data we follow the steps described by (Junczys-Dowmunt and Grundkiewicz, 2016). Next we train a English-to-German MT system using data from the WMT2016 shared task on IT translation. This system is used to translate it's own training data into German. Although input sentence have been seen, the translations are far from perfect. Next we create an MT system to translate from correct German to imperfect German MT output. This system can now be applied to create raw German MT output from correct German text.

In order to achieve matching TER statistics we use a simple implementation of the Nelder-Mead algorithm for parameter tuning. For unknown reasons, MERT or kb-Mira would not create output with the desired error-rates.

Using this system we create a new large set of pseudo-PE data, translating domain-selected monolingual data from German into German pseudo-MT output. The English input is created with an German-to-English phrase-based MT system. We translate about 15 million sentences in this manner, creating new artificial APE triplets.

4 Experiments and Results

4.1 Training, Development, and Test Data

We perform all our experiments with the official WMT16 (Bojar et al., 2016) automatic post-

	dev 2016		test 2016		test 2017	
Model	**TER**↓	BLEU↑	**TER**↓	BLEU↑	**TER**↓	BLEU↑
WMT17-baseline 1	–	–	–	–	24.48	62.49
WMT17-baseline 2	–	–	–	–	24.69	62.97
CONTRASTIVE	19.74	70.61	19.30	70.34	19.83	69.38
PRIMARY	–	–	19.21	70.51	19.77	69.50

Table 2: Submitted system results

editing data and the respective development and test sets. The training data consists of a small set of 23,000 post-editing triplets (src, mt, pe), where src is the original English text, mt is the raw MT output generated by an English-to-German system, and pe is the human post-edited MT output. The MT system used to produce the raw MT output is unknown, so is the original training data. The task consist of automatically correcting the MT output so that it resembles human post-edited data. The main task metric is TER (Snover et al., 2006) – the lower the better – with BLEU (Papineni et al., 2002) as a secondary metric.

Table 1 summarizes the data sets used in this work. To produce our final training data set we oversample the original training data 20 times and add all three artificial data sets (they may overlap). This results in a total of slightly more than 21M training triplets. We keep the development set as a validation set for early stopping and report results on the WMT16 test set. The data is already tokenized, additionally we truecase all files and apply segmentation into BPE subword units. We reuse the subword units distributed with the artificial data set. For the hard-attention models, we create new target training and development files following the procedure from section 2.2.

4.2 Training parameters

All models are trained on the same training data. Models with single input encoders take only the raw MT output (mt) as input, double-encoder models use raw MT output (mt) and the original source (pe). The training procedures and model settings are the same whenever possible:

- All embedding vectors consist of 512 units, the RNN states use 1024 units. We choose a vocabulary size of 40,000 for all inputs and outputs. When hard attention models are trained the maximum sentence length is 100 to accommodate the additional step symbols, otherwise 50.
- To avoid overfitting, we use pervasive dropout (Gal, 2015) over GRU steps and input embeddings, with dropout probabilities 0.2, and over source and target words with probabilities 0.2.
- We use Adam (Kingma and Ba, 2014) as our optimizer, with a mini-batch size of 64. All models are trained with Asynchronous SGD (Adam) on three to four GPUs.
- We train all models until convergence (early-stopping with a patience of 10 based on dev-set cross-entropy cost), saving model checkpoints every 10,000 mini-batches.
- The best eight model checkpoints w.r.t. dev-set cross-entropy of each training run are averaged element-wise (Junczys-Dowmunt et al., 2016) resulting in new single models with generally improved performance.
- For the multi-source models we repeat the mentioned procedure four times with different randomly initialized weights and random seeds to later form model ensembles.

Training time for one model on four NVIDIA GTX 1080 GPUs or NVIDIA TITAN X (Pascal) GPUs is between two and three days, depending on model complexity.

4.3 Submitted System

We chose an ensemble of four ENCDEC-HARD+DOUBLE-ATT systems (four distinct training runs with different random weights initializations) as our final system. In Table 2, this system is marked as CONSTRASTIVE. We also noticed that providing the system output once more as system input to the same system results in a small im-

Model	dev 2016 TER↓	dev 2016 BLEU↑	test 2016 TER↓	test 2016 BLEU↑
WMT16-baseline 1 (Bojar et al., 2016)	25.14	62.92	24.76	62.11
WMT16-baseline 2 (Bojar et al., 2016)	–	–	24.64	63.47
Junczys-Dowmunt and Grundkiewicz (2016)	21.46	68.94	21.52	67.65
Pal et al. (2017) SYMMETRIC	–	–	21.07	67.87
Pal et al. (2017) RERANKING	–	–	20.70	**69.90**

Table 3: Results from the literature for the WMT 2016 APE development and test set

Model	dev 2016 TER↓	dev 2016 BLEU↑	test 2016 TER↓	test 2016 BLEU↑
ENCDEC-ATT	22.01	68.11	22.27	66.90
ENCDEC-HARD	22.72	66.82	22.72	65.86
ENCDEC-HARD+ATT	22.11	67.82	22.10	67.15
ENCDEC-DOUBLE-ATT	20.79	69.28	20.69	68.56
ENCDEC-DOUBLE-ATT × 4	20.10	**70.24**	**19.92**	69.40
ENCDEC-HARD+DOUBLE-ATT	20.83	69.02	20.87	68.14
ENCDEC-HARD+DOUBLE-ATT × 4	**20.08**	70.05	20.34	68.96

Table 4: Post-submission results, the main task metric is TER (the lower the better)

provement. This one-time looped system is our primary submission PRIMARY.

5 Post-submission analysis

This section is based on the work in Junczys-Dowmunt and Grundkiewicz (2017). After the submission we performed a number of in-depth experiments to verify our intuitions about the selected models for a better controlled data setting. We restricted all training, development data to data available during the WMT 2016 shared task on APE and test on test set 2016. We also only used artificial data made available by Junczys-Dowmunt and Grundkiewicz (2016), dicarding the newly created data in this work. To produce our final training data set we oversample the original training data 20 times and add the artificial data sets. This results in a total of slightly more than 5M training triplets. For the hard-attention models, we create new target training and development files following the LCS-based procedure outlined in section 2.2.

Table 3 contains a selection of most relevant results for the WMT16 APE shared task – during the task and afterwards. WMT 2016-baseline 1 is the raw uncorrected mt output, baseline 2 is the results of a vanilla phrase-based Moses system (Koehn et al., 2007) trained only on the official 12,000 sentences. Junczys-Dowmunt and Grundkiewicz (2016) is the best system at the shared task. Pal et al. (2017) SYMMETRIC is the currently best reported result on the WMT16 APE test set for a single neural model (single source), whereas Pal et al. (2017) RERANKING – the overall best reported result on the test set – is a system combination of Pal et al. (2017) SYMMETRIC with phrase-based models via n-best list re-ranking.

In Table 4 we present the results for the models discussed in this work. The double-attention models outperform the best WMT16 system and the currently reported best single-model Pal et al. (2017) SYMMETRIC. The ensembles also beat the system combination Pal et al. (2017) RERANKING in terms of TER (not in terms of BLEU though). The simpler double-attention model with no hard-attention ENCDEC-DOUBLE-ATT reaches slightly better results on the test set than its counterpart with added hard attention ENCDEC-HARD+DOUBLE-ATT, but the situation would have been less clear if only the dev set were used to choose the best model.

Acknowledgments

This research was funded by the Amazon Academic Research Awards program.

References

Roee Aharoni and Yoav Goldberg. 2016. Sequence to sequence transduction with hard monotonic attention. *arXiv preprint arXiv:1611.01487* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 131–198. http://www.aclweb.org/anthology/W/W16/W16-2301.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Doubly-attentive decoder for multi-modal neural machine translation. *CoRR* abs/1702.01287. http://arxiv.org/abs/1702.01287.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc. of Empirical Methods in Natural Language Processing*.

Yarin Gal. 2015. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *ArXiv e-prints* .

Daniel S. Hirschberg. 1977. Algorithms for the longest common subsequence problem. *J. ACM* 24(4):664–675. https://doi.org/10.1145/322033.322044.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? A case study on 30 translation directions. *arXiv preprint arXiv:1610.01108* http://arxiv.org/abs/1610.01108.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation*. pages 751–758.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2017. An exploration of neural sequence-to-sequence architectures for automatic post-editing. *CoRR* abs/1706.04138. https://arxiv.org/abs/1706.04138.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* http://arxiv.org/abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 177–180.

Jindrich Libovický and Jindrich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. *CoRR* abs/1704.06567. http://arxiv.org/abs/1704.06567.

Jindřich Libovický, Jindřich Helcl, Marek Tlustý, Ondřej Bojar, and Pavel Pecina. 2016. CUNI system for WMT16 automatic post-editing and multimodal translation tasks. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 646–654. http://www.aclweb.org/anthology/W/W16/W16-2361.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the Annual Meeting on Association for Computational Linguistics*. pages 160–167.

Santanu Pal, Sudip Kumar Naskar, Mihaela Vela, Qun Liu, and Josef van Genabith. 2017. Neural automatic post-editing using prior alignment and reranking. In *Proceedings of the European Chapter of the Association for Computational Linguistics*. pages 349–355.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain, pages 65–68. http://aclweb.org/anthology/E17-3017.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas,*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *CoRR* abs/1601.00710. http://arxiv.org/abs/1601.00710.