# CUNI System for WMT17 Automatic Post-Editing Task

**Dušan Variš and Ondřej Bojar**
Charles University, Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics,
Malostranské náměstí 25, 118 00 Prague, Czech Republic
{varis,bojar}@ufal.mff.cuni.cz

## Abstract

Following upon the last year's CUNI system for automatic post-editing of machine translation output, we focus on exploiting the potential of sequence-to-sequence neural models for this task. In this system description paper, we compare several encoder-decoder architectures on a smaller-scale models and present the system we submitted to WMT 2017 Automatic Post-Editing shared task based on this preliminary comparison. We also show how simple inclusion of synthetic data can improve the overall performance as measured by an automatic evaluation metric. Lastly, we list few example outputs generated by our post-editing system.

## 1 Introduction

Even with the recent substantial improvements of the machine translation (MT) quality mainly thanks to the increasingly popular neural models (neural MT, NMT), many errors still remain in the output require further post-editing. This can be done manually, or as the automatic post-editing (APE) task expects, automatically.

When phrase-based machine translation (PBMT) was the indisputable state of the art, some automatic post-editing (APE) systems were based on the PBMT techniques (Simard et al., 2007). With source-sentence information (Béchara et al., 2011), post-editing results were quite promising. It is therefore not surprising that with the rise of the neural machine translation, neural APE systems based on the findings in NMT research were built (Pal et al., 2016) and even won last year's WMT16 Shared Task (Junczys-Dowmunt and Grundkiewicz, 2016).

In this paper, we present a baseline comparison of several recent neural sequence-to-sequence architectures, motivations behind our primary submission for the WMT17 Shared Task and further improvements of this submission with regard to model size and additional synthetic data.

## 2 Experiments

In automatic post-editing, we are expected to take the output of an MT system that usually contains various errors (morphological, lexical etc.) and to generate a corrected version of the output. Most of the time, there is also additional information available, e.g. the original sentence in the source language and sometimes also some internal scores or features from the primary MT system.

### 2.1 Examined Setups

If we look at the recent developments in the field of NMT, we can see that there are many different novel approaches that often bring significant improvements to the overall performance of the NMT system. It is natural to ask how these findings can be applied to APE task and how much they can contribute to the APE system performance. We experimented in two areas: (1) how to feed simultaneously the source sentence and the MT output (multi-source input), and (2) whether to use subword units or individual characters.

#### 2.1.1 Multi-Source Input

All our experiments use both the source sentence and the MT output to be corrected. As far as encoding the input is concerned, we examined two basic approaches. We tried using a single encoder that received the concatenation of the source sentence and the corresponding MT output as suggested by Niehues et al. (2016). The resulting input sequence becomes longer and it may thus be more difficult to encode, but it was reported that

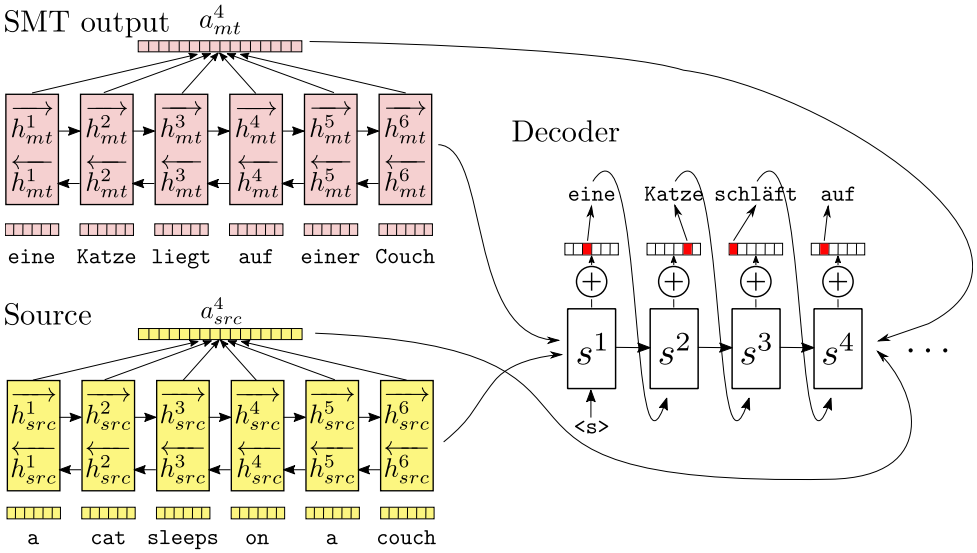


Figure 1: Illustration of a multiple-encoder sequence-to-sequence architecture as illustrated in Libovický et al. (2016).

(through the attention mechanism) the decoder is able to attend to relevant parts of the concatenated sentences when generating output.

As an alternative option, we also tried using two separate encoders, one for the source sentence and one for the MT output (Libovický et al., 2016) as shown in Figure 1. In this case, both encoders encode their corresponding input sequences separately and the concatenation of their final states is passed to the decoder. The attention is computed over the hidden states of both encoders as if they were produced by a single encoder. Libovický and Helcl (2017) present other options for combining the attention of multiple encoders, but the investigation of these methods is not covered in this paper.

#### 2.1.2 Subword Units or Characters

All data-driven approaches to MT suffer in quality when translating rare words (including words not seen during training at all) and NMT is no exception. In out neural approach to APE, we would still like our APE system to address errors in rare words (e.g. by fixing their endings). A popular approach of reducing the vocabulary size in NMT is called byte-pair encoding (BPE, Sennrich et al., 2015) which creates a vocabulary of most frequent words, subword units and individual characters. This way, even rare words can be successfully handled by modifying their parts.

Another option is to use a fully character-level encoder-decoder architecture. However, this approach in its basic form results in much longer sequences that are generally much harder to learn for the underlying recurrent neural network (RNN, Pascanu et al., 2012). Another downside is the increased training and inference time for each sentence. Recently, Lee et al. (2016) presented an encoder architecture that uses RNN over the output of several hundreds convolutional filters that are applied on the character-level embeddings, combining the benefits of both convolutional and recurrent approaches.

### 2.2 Baseline Comparison

Based on the approaches described in the previous section, we decided to compare the following system variations:

- a single encoder (concatenated input, "concat") vs. two separate encoders ("two-enc"),
- BPE2BPE vs. CHAR2CHAR architecture.

Each system variation was trained using a single Nvidia Tesla K20 5GB GPU. We set embedding size and both encoder and decoder RNN size to 300 for all the systems. We used BPE vocabulary of size 50k for the BPE2BPE systems and character vocabulary of size 500 for the CHAR2CHAR systems. We did not use dropout during training. For the CHAR2CHAR setups (i.e. RNN over convolutional encoder by Lee et al., 2016), we reduced the number of convolutional filters proportionally to the size of the used GPU, used segment size 5 and highway network of depth 1.

| System | BLEU |
|---|---|
| BPE2BPE two-enc | 42.36 |
| BPE2BPE concat | 42.13 |
| CHAR2CHAR two-enc | 49.82 |
| CHAR2CHAR concat | 49.94 |

Table 1: Automatic evaluation of the proposed architectures we trained. The model size was downscaled to 5GB due to the limited computation resources.

The experiments were carried out in Neural Monkey[1] (Helcl and Libovický, 2017), a framework for sequence-to-sequence modeling. Most of the required neural network components together with necessary preprocessing and postprocessing were already implemented in the framework. We added the RNN over convolutional encoder in this work.

We used 12k sentences WMT16 APE training dataset for training and we computed BLEU (Papineni et al., 2002) on WMT16 APE development dataset to compare the baselines. The evaluation was performed during training. We thus did not use beam search and simply greedily chose the most probable output at each decoding step to get the validation output.

The best results for each architecture are shown in Table 1. We can see that the character-level post-editing models outperform the subword-level models. However, the training was done using only a small dataset which may possibly indicate that the character level architecture is able to better exploit the training data. Nevertheless, we chose the character-level system for our remaining experiments.

## 3 CUNI System for WMT17 APE Task

After the baseline comparison of the smaller sequence-to-sequence models, we moved towards training of the primary submission for the WMT17 post-editing task.

### 3.1 Common Settings

We decided to use the two-encoder character-level architecture. Even though the single encoder character-level architecture with concatenated inputs performed slightly better during the baseline evaluation, we believe that the multi-encoder architecture offers higher potential for further improvement.[2]

The model was trained using GeForce GTX 1080 with 8GB memory with the following parameters:

- shared character-level vocabulary size: 500
- encoder RNN size: 256
- input embedding size: 300
- segment size: 5
- highway network depth: 2
- convolutional filters (size, number of filters): (1,150), (2,200), (3,250), (4,250), (5,300), (6,300), (7,350), (8,350)
- decoder RNN size: 512
- output embedding size: 300

During the inference, we used beam-search of beam size 20 and length normalization to penalize shorter sentences. Beam search parameters were chosen based on the Lee et al. (2016).

First, we used only 23k sentences from WMT17 training dataset to train the system. We used this model as a baseline which we tried to further improve.

### 3.2 Synthetic Data

Since the basic training dataset provided for the task was rather small we also tried to include the training dataset from the previous WMT16 post-editing task and furthermore, we added the synthetic data (smaller dataset, ∼500k sentences) as provided by last year's submission of Junczys-Dowmunt and Grundkiewicz (2016). To balance the ratio of genuine and synthetic sentences in the final dataset, we duplicated the WMT16 and WMT17 sentence pairs several times to match the size of the synthetic dataset. We then took all the data and shuffled them randomly to create a dataset consisting of ∼1M training sentences. We used WMT16 APE dev set to evaluate the model during the training.

[1] https://github.com/ufal/neuralmonkey

[2] This still needs to be confirmed though by the future research.

| | |
|---|---|
| Source | You can also perform many types of transformations by dragging the bounding box for a selection . |
| OrigMT | Sie können auch zahlreiche Transformationsarten durchführen , indem Sie den Begrenzungsrahmen für eine Auswahl . |
| Synth | Sie können auch zahlreiche Transformationsarten durchführen , indem Sie den Begrenzungsrahmen für eine Auswahl **ziehen** . |
| Ref | Sie können auch zahlreiche Transformationsarten durchführen , indem Sie den Begrenzungsrahmen für eine Auswahl ziehen . |
| Source | 3D comments added to other views are listed as components of that view in the Model Tree . |
| OrigMT | 3D hinzugefügten Kommentare zu anderen Ansichten als Komponenten anzuzeigen , die in der Modellhierarchie aufgeführt sind . |
| Synth | 3D-Kommentare zu anderen Ansichten **werden** als Komponenten **angezeigt** , die in der Modellhierarchie aufgeführt sind . |
| Ref | Anderen Ansichten hinzugefügte 3D-Kommentare werden in der Modellhierarchie als Komponenten dieser Ansicht aufgeführt . |
| Source | Choose an option from the Key Algorithm menu . |
| OrigMT | Wählen Sie eine Option aus dem Menü " Algorithm . " |
| Synth | Wählen Sie eine Option aus dem Menü " Algorithmu . " |
| Ref | Wählen Sie eine Option aus dem Menü " Schlüsselalgorithmus . " |
| Source | Shift-drag to constrain the movement of the object horizontally , vertically , or diagonally . |
| OrigMT | Halten Sie beim Ziehen des Zeigers über die Bewegung des Objekts horizontal , vertikal oder diagonal einzuschränken . |
| Synth | Halten Sie beim Ziehen die Bewegung des Objekts die Bewegung des Objekts horizontal , vertikal oder diagonal eingeschränkt . |
| Ref | Halten Sie beim Ziehen des Objekts die Umschalttaste gedrückt , um nur horizontale , vertikale oder diagonale Bewegungen zuzulassen . |

Figure 2: Sample outputs from the original MT and our submitted model "Synth". In the first two examples, our model helped to produce correctly the main verb (in **bold**). In the third example, it introduced a spelling error (underlined). The last example shows that the model can also severely damage the sentence, introducing repetitions common in NMT output. The original output for the last sentence was not perfect either, it does not mention the shift key at all (and our model does not fix it).

### 3.3 Predicting Edit Operations

Finally, inspired by Libovický et al. (2016), we also trained a separate model that generates a sequence of post-editing operations ("editops") instead of directly generating the target sequence of characters. Aside from generating characters present in the training data, the model learns to use special tokens "<keep>" and "<delete>", or to normally produce characters present in the training data, to indicate the modifications needed for the MT output. We used the same network parameters and data (including the synthetic dataset) for the model with and without BPE.

### 3.4 Evaluation

We evaluated these three models using the WMT16 APE test set[3], computing the BLEU score on the produced outputs: baseline CHAR2CHAR setup (Baseline), the model trained with synthetic data (Synth) and the model which produces edit operations instead of complete sentences (Synth+editops). Table 2 shows the results of the evaluation.

We can see that even when we choose the best architecture based on the relative comparison and increase the model capacity ("Baseline"), it is still not enough to even get close to the original MT output quality ("Original MT"). Introducing additional synthetic data ("Synth") fixed this and actually outperformed the original MT, reaching BLEU of 66.04. We chose this system as our primary submission for the WMT16 APE task.

| System | BLEU |
|---|---|
| Original MT | 62.09 ($\pm$1.04) |
| Baseline | 50.86 ($\pm$3.96) |
| Synth | **66.04 ($\pm$1.16)** |
| Synth+editops | 62.08 ($\pm$1.05) |

Table 2: Automatic evaluation of the final 8GB APE setups. The score of the original MT output is shown for comparison. The $\pm$ values are empirical confidence intervals reflecting the variance in the test set (Koehn, 2004).

We were a little surprised that there was no improvement when using model that learned to generate post-editing operations ("Synth+editops"). When we manually examined the generated output, we found out that the system took the safer path of keeping most of the machine translation output because it probably resulted in fewer errors than trying to change it. This could be probably avoided by discouraging the model from keeping the whole MT output unchanged and we plan investigating this approach in the future.

Even though we did not perform a thorough manual evaluation, we present some examples of our submitted system ("Synth") outputs to give the reader some insight to the model performance in Figure 2. Our post-editing helped with the main verb, but in other cases, it also damaged the sentence structure or introduced spelling errors.

[3] http://www.statmt.org/wmt16/ape-task.html

## 4 Conclusion

In this paper, we compared several sequence-to-sequence architectures that were previously proposed for the NMT task and evaluated their performance in automatic post-editing of English-to-German MT output. Our setup relies on the original source sentence and uses either subword units (BPE) or individual characters.

With additional synthetic data, we were able to improve over the original MT output in terms of BLEU, but a quick manual inspection reveals that errors can be easily also introduced and BLEU (or other automatic metric) is not likely to give a reliable picture of the post-editing performance.

## Acknowledgments

This work has been in part supported by the EU grants no. H2020-ICT-2014-1-644402 (Health in my Language) and H2020-ICT-2014-1-645452 (QT21), as well as by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071) and Charles University SVV project no. 260 453.

Computational resources were also supplied by the Ministry of Education, Youth and Sports of the Czech Republic under the Projects CESNET (Project No. LM2015042), CERIT-Scientific Cloud (Project No. LM2015085) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

## References

Hanna Béchara, Yanjun Ma, and Josef van Genabith. 2011. Statistical Post-Editing for a Statistical MT System. In *Proceedings of the 13th Machine Translation Summit*. pages 308–315.

Jindřich Helcl and Jindřich Libovický. 2017. Neural monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics* (107):5–17. https://doi.org/10.1515/pralin-2017-0001.

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, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*. The Association for Computer Linguistics, pages 751–758. http://aclweb.org/anthology/W/W16/W16-2378.pdf.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of EMNLP 2004*. Barcelona, Spain.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *CoRR* abs/1610.03017. http://arxiv.org/abs/1610.03017.

Jindřich Libovický and Jindřich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada.

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/W16-2361.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-translation for neural machine translation. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. ACL, pages 1828–1836. http://aclweb.org/anthology/C/C16/C16-1172.pdf.

Santanu Pal, Sudip Kumar Naskar, Mihaela Vela, and Josef van Genabith. 2016. A neural network based approach to automatic post-editing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics. http://aclweb.org/anthology/P/P16/P16-2046.pdf.

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.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR* abs/1211.5063. http://arxiv.org/abs/1211.5063.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR* abs/1508.07909. http://arxiv.org/abs/1508.07909.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical phrase-based post-editing. In

*Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Association for Computational Linguistics, Rochester, New York, pages 508–515. http://www.aclweb.org/anthology/N/N07/N07-1064.