Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing

Marcin Junczys-Dowmunt and **Roman Grundkiewicz**
Adam Mickiewicz University in Poznań
ul. Umultowska 87, 61-614 Poznań, Poland
{junczys,romang}@amu.edu.pl

Abstract

This paper describes the submission of the AMU (Adam Mickiewicz University) team to the Automatic Post-Editing (APE) task of WMT 2016. We explore the application of neural translation models to the APE problem and achieve good results by treating different models as components in a log-linear model, allowing for multiple inputs (the MT-output and the source) that are decoded to the same target language (post-edited translations). A simple string-matching penalty integrated within the log-linear model is used to control for higher faithfulness with regard to the raw machine translation output. To overcome the problem of too little training data, we generate large amounts of artificial data. Our submission improves over the uncorrected baseline on the unseen test set by -3.2% TER and +5.5% BLEU and outperforms any other system submitted to the shared-task by a large margin.

1 Introduction

This paper describes the submission of the AMU (Adam Mickiewicz University) team to the Automatic Post-Editing (APE) task of WMT 2016. Following the APE shared task from WMT 2015 (Bojar et al., 2015), the aim is to test methods for correcting errors produced by an unknown machine translation system in a black-box scenario. The organizers provide training data with human post-edits, evaluation is carried out part-automatically using TER (Snover et al., 2006) and BLEU (Papineni et al., 2002), and part-manually.

We explore the application of neural translation models to the APE task and investigate a number of aspects that seem to lead to good results:

- Creation of artificial post-edition data that can be used to train the neural models;
- Log-linear combination of monolingual and bilingual models in an ensemble-like manner;
- Addition of task-specific features in a log-linear model that allow to control for faithfulness of the automatic post-editing output with regard to the input, otherwise a weakness of neural translation models.

According to the automatic evaluation metrics used for the task, our system is ranked first among all submission to the shared task.

2 Related work

2.1 Post-Editing

State-of-the-art APE systems follow a monolingual approach firstly proposed by Simard et al. (2007) who trained a phrase-based SMT system on machine translation output and its post-edited versions. Béchara et al. (2011) proposed a “source-context aware” variant of this approach: automatically created word alignments are used to create a new source language which consists of joined MT-output and source token pairs. The inclusion of source-language information in that form is shown to be useful to improve the automatic post-editing results (Béchara et al., 2012; Chatterjee et al., 2015b). The quality of the word alignments plays an important role for this methods, as shown for instance by Pal et al. (2015).

A number of techniques have been developed to improve PB-SMT-based APE systems, e.g. approaches relying on phrase-table filtering techniques and specialized features. Chatterjee et al. (2015a) propose a pipeline where the best language model and pruned phrase table are selected through task-specific dense features. The goal was to overcome data sparsity issues.

The authors of the Abu-MaTran system (no publication, see Bojar et al. (2015)) incorporate sentence-level classifiers in a post-processing step which choose between the given MT output or an automatic post-edition coming from a PB-SMT APE system. Their most promising approach consists of a word-level recurrent neural network sequence-to-sequence classifier that marks each word of a sentence as *good* or *bad*. The output with the lower number of *bad* words is then chosen as the final post-editing answer. We believe this work to be among the first to apply (recurrent) neural networks to the task of automatic post-editing.

Other popular approaches rely on rule-based components (Wisniewski et al., 2015; Béchara et al., 2012) which we do not discuss here.

2.2 Neural machine translation

We restrict our description to the recently popular encoder-decoder models, based on recurrent neural networks (RNN).

An LSTM-based encoder-decoder model was introduced by Sutskever et al. (2014). Here the source sentence is encoded into a single continuous vector, the final state of the source LSTM-RNN. Once the end-of-sentence marker has been encoded, the network generates a translation by sampling the most probable translations from the target LSTM-RNN which keeps its state based on previous words and the source sentence state.

Bahdanau et al. (2015) extended this simple concept with bidirectional source RNNs (Cho et al., 2014) and the so-called soft-attention model. The novelty of this approach and its improved performance compared to Sutskever et al. (2014) came from the reduced reliance on the source sentence embedding which had to convey all information required for translation in a single state. Instead, attention models learn to look at particular word states at any position within the source sentence. This makes it also easier for these models to learn when to make copies, an important aspect for APE. We refer the reader to Bahdanau et al. (2015) for a detailed description of the discussed models. At the time of writing, no APE systems relying on neural translation models seem to have been published.[1]

3 Data and data preparation

3.1 Used corpora

It was explicitly permitted to use additional data while preparing systems for the APE shared task. We made use of the following resources:

1. The official training and development data provided by the APE shared task organizers, consisting of 12,000 training triplets[2] and 1,000 development set triplets. In this paper we report our results for the 1,000 sentences of development data, and selected results on the unseen test data as provided by the task organizers.

2. The domain-specific English-German bilingual training data admissible during the WMT-16 shared task on IT-domain translation;

3. All other parallel English-German bilingual data admissible during the WMT-16 news translation task;

4. The German monolingual Common Crawl corpus admissible for the WMT-16 news translation and IT translation tasks.

3.2 Pre- and post-processing

The provided triplets have already been tokenized, the tokenization scheme seems to correspond to the Moses (Koehn et al., 2007) tokenizer without escaped special characters, so we re-apply escaping. All other data is tokenized with the Moses tokenizer with standard settings per language. We truecase the data with the Moses truecaser.

To deal with the limited ability of neural translation models to handle out-of-vocabulary words we split tokens into subword units, following Sennrich et al. (2015b).

Subword units were learned using a modified version of the byte pair encoding (BPE) compression algorithm (Gage, 1994). Sennrich et al. (2015b) modified the algorithm to work on character level instead of on bytes. The most frequent pairs of characters are iteratively replaced by a new character sequence created by merging the pairs of existent sequences. Frequent words

[1]An accepted ACL 2016 paper is scheduled to appear: Santanu Pal, Sudip Kumar Naskar, Mihaela Vela and Josef van Genabith. A Neural Network based Approach to Automated Post-Editing. Proceedings of the 54th Annual Meetings of the Association for Computational Linguistics, August 2016.

[2]A triplet consists of the English source sentence, a German machine translation output, and the German manually post-edited correction of that output.

are thus represented by single symbols and infrequent ones are divided into smaller units. The final size of the vocabulary is equal to the sum of merge operations and the number of initial characters. This method effectively reduces the number of unknown words to zero, as characters are always available as the smallest fall-back units. Sennrich et al. (2015b) showed that this method can deal with German compound nouns (relieving us from applying special methods to handle these) as well as transliterations for Russian-English.

This seems particularly useful in the case of APE, where we do not wish the neural models to "hallucinate" output when encountering unknown tokens. A faithful transliteration is more desirable. We chose vocabularies of 40,000 units per language. For German MT output and post-edited sentences we used the same set of subword units.

4 Artificial post-editing data

The provided post-editing data is orders of magnitude too small to train our neural models, and even with the in-domain training data from the IT translation task, we quickly see overfitting effects for a first English-German translation system. Inspired by Sennrich et al. (2015a) — who use back-translated monolingual data to enrich bilingual training corpora — we decided to create artificial training triplets.

4.1 Bootstrapping monolingual data

We applied cross-entropy filtering (Moore and Lewis, 2010) to the German Common Crawl corpus performing the following steps:

- We filtered the corpus for "well-formed" lines which start with a capital Unicode letter character and end in an end-of-sentence punctuation mark. We require the line to contain at least 30 Unicode letters.
- The corpus has been preprocessed as described above, including subword units, which may have a positive effect on cross-entropy filtering as they allow to score unknown words.
- Next, we built an in-domain trigram language model (Heafield et al., 2013) from the German post-editing training data and the German IT-task data, and a similarly sized out-of-domain language model from the Common Crawl data.
- We calculated cross-entropy scores for the first one billion lines of the corpus according to the two language models;
- We sorted the corpus by increasing cross-entropy and kept the first 10 million entries for round-trip translation and the top 100 million entries for language modeling.

4.2 Round-trip translation

For the next step, two phrase-based translation models, English-German and German-English, were created using the admissible parallel training data from the IT task. Word-alignments were computed with fast-align (Dyer et al., 2013), the dynamic-suffix array (Germann, 2015) holds the translation model. The top 10% bootstrapped monolingual data was used for language modeling in case of the English-German model, for the German-English translation system the language model was built only from the target side of the parallel in-domain corpora.[3]

The top 1% of the bootstrapped data have first been translated from German to English and next backwards from English to German. The intermediate English translations were preserved. In order to translate these 10 million sentences quickly (twice), we applied small stack-sizes and cube-pruning-pop-limits of around 100, completing the round-trip translation in about 24 hours.

This procedure left us with 10 million artificial post-editing triplets, where the source German data is treated as post-edited data, the German$\rightarrow$English translated data is the English source, the round-trip translation results are the new uncorrected MT-output.

4.3 Filtering for TER

We hope that a round-trip translation process produces literal translations that may be more-or-less similar to post-edited triplets, where the distance between MT-output and post-edited text is generally smaller than between MT-output and human-produced translations of the same source. Having that much data available, we could continue our filtering process by trying to mimic the TER-statistics of the provided APE training corpus. While TER scores do only take into account the two German language parts of the triplet, it

[3]These models were not meant to be state-of-the-art quality systems. Our main objective was to create them within a few hours.

Data set	Sentences	NumWd	WdSh	NumEr	TER
training set	12,000	17.89	0.72	4.69	26.22
development set	1,000	19.76	0.71	4.90	24.81
round-trip.full	9,960,000	13.50	0.58	5,72	42.02
round-trip.n10	4,335,715	15.86	0.66	5.93	36.63
round-trip.n1	531,839	20.92	0.55	5.20	25.28

Table 1: Statistics of full and filtered data sets: number of sentences, average number of words, word shifts, errors, and TER score.

seems reasonable that filtering for better German-German pairs automatically results in a higher quality of the intermediate English part.

To achieve this, we represented each triplet in the APE training data as a vector of elementary TER statistics computed for the MT-output and the post-edited correction, such as the sentence length, the frequency of edit operations, and the sentence-level TER score. We do the same for the to-be-filtered artificial triplet corpus. The similarity measure is the inverse Euclidean distance over these vector representations.

In a first step, outliers which diverge from any maximum or minimum value of the reference vectors by more than 10% were removed. For example, we filtered triplets with post-edited sentences that were 10% longer than the longest post-edited sentence in the reference.

In the second step, for each triplet from the reference set we select n nearest neighbors. Candidates that have been chosen for one reference set triplet were excluded for the following triplets. If more than the 100 triplets had to be traversed to satisfy the exclusion criterion, less than n or even 0 candidates were selected. Two subsets have been created, one for $n = 1$ and one for $n = 10$. Table 1 sets the characteristics of the obtained corpora in relation to the provided training and development data. The smaller set (round-trip.n1) follows the TER statistics of the provided training and development data quite closely, but consists only of 5% of the artificial triplets. The larger set (round-trip.n10) consists of roughly 43% of the data, but has weaker TER scores.

5 Experiments

Following the post-editing-by-machine-translation paradigm, we explore the application of soft-attention neural translation models to post-editing. Analogous to the two dominating approaches described in Section 2.1, we investigate methods that are purely monolingual as well as a simple method to include source language information in a more natural way than it has been done for phrase-based machine translation.

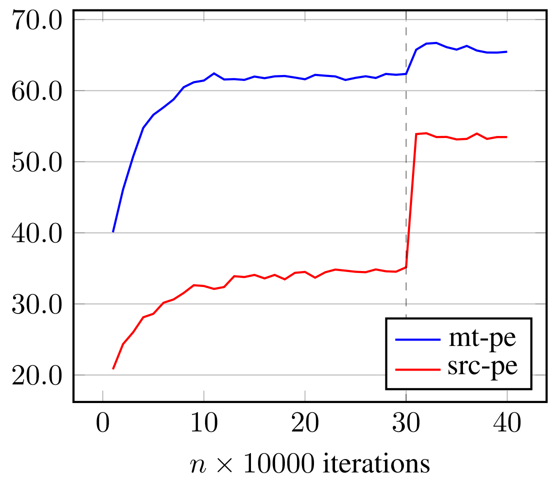

Figure 1: Training progress for mt-pe and src-pe models according to development set; dashed vertical line marks change from training set round-trip.n10 to fine-tuning with round-trip.n1.

The neural machine translation systems explored in this work are attentional encoder-decoder models (Bahdanau et al., 2015), which have been trained with Nematus[4]. We used mini-batches of size 80, a maximum sentence length of 50, word embeddings of size 500, and hidden layers of size 1024. Models were trained with Adadelta (Zeiler, 2012), reshuffling the corpus between epochs. As mentioned before tokens were split into subword units, 40,000 per language. For decoding, we used AmuNMT[5], our C++/CUDA decoder for NMT models trained with Nematus with a beam size of 12 and length normalization.

[4]https://github.com/rsennrich/nematus
[5]https://github.com/emjotde/amunmt

System	TER	BLEU
Baseline (mt)	25.14	62.92
mt→pe	23.37	66.71
mt→pe×4	23.23	66.88
src→pe	32.31	53.89
src→pe×4	31.42	55.41
mt→pe×4 / src→pe×4*	22.38	68.07
mt→pe×4 / src→pe×4 / pep*	**21.46**	**68.94**

Table 2: Results on provided development set. Best-performing models have been chosen based on this development set. Systems marked with * have weights tuned on the same development set.

5.1 MT-output to post-editing

We started training the monolingual MT-PE model with the MT and PE data from the larger artificial triplet corpus (round-trip.n10). The model has been trained for 4 days, saving a model every $10,000$ mini-batches. Quick convergence can be observed for the monolingual task and we switched to fine-tuning after the 300,000-th iteration with a mix of the provided training data and the smaller round-trip.n1 corpus. The original post-editing data was oversampled 20 times and concatenated with round-trip.n1.

This resulted in the performance jump shown in Figure 1 (mt→pe, blue). Training were continued for another 100,000 iterations and stopped when overfitting effects became apparent. Training directly with the smaller training data without the initial training on round-trip.n10 lead to even earlier overfitting.

Entry mt→pe in Table 2 contains the results of the single-best model on the development set which outperforms the baseline significantly. Models for ensembling are selected among the periodically saved parameter dumps of one training run. An ensemble mt→pe×4 consisting of the four best models shows only modest improvements over the single model. The same development set has been used to select the best-performing models, results may therefore be slightly skewed.

5.2 Source to post-editing

We proceed similarly for the English-German NMT training. When fine-tuning with the smaller corpus with oversampled post-editing data, we also add all in-domain parallel training data from the IT-task, roughly 200,000 sentences. Fine-tuning results in a much larger jump than in the monolingual case, but the overall performance of the NMT system is still weaker than the uncorrected MT-baseline.

As for the monolingual case, we evaluate the single-best model (src→pe) and an ensemble (src→pe×4) of the four best models of a training run. The src→pe×4 system is not able to beat the MT baseline, but the ensemble is significantly better than the single model.

5.3 Log-linear combinations and tuning

AmuNMT can be configured to accept different inputs to different members of a model ensemble as long as the target language vocabulary is the same. We can therefore build a decoder that takes both, German MT output and the English source sentence, as parallel input, and produces post-edited German as output. Since once the input sentence has been provided to a NMT model it essentially turns into a language model, this can be achieved without much effort. In theory an unlimited number of inputs can be combined in this way without the need of specialized multi-input training procedures (Zoph and Knight, 2016).[6]

In NMT ensembles, homogeneous models are typically weighted equally. Here we combine different models and equal weighting does not work. Instead, we treat each ensemble component as a feature in a traditional log-linear model and perform weighting as parameter tuning with Batch-Mira (Cherry and Foster, 2012). AmuNMT can produce Moses-compatible n-best lists and we devised an iterative optimization process similar to the one available in Moses. We tune the weights on the development set towards lower TER scores; two iterations seem to be enough. When ensembling one mt→pe model and one src→pe model, the assigned weights correspond roughly to 0.8 and 0.2 respectively. The linear combination of all eight models (mt→pe×4 / src→pe×4) improves quality by 0.9 TER and 1.2 BLEU, however, weights were tuned on the same data.

5.4 Enforcing faithfulness

We extend AmuNMT with a simple Post-Editing Penalty (PEP). To ensure that the system is fairly

[6]Which are still worth investigating for APE and likely to yield better results.

conservative — i.e. the correction process does not introduce too much new material — every word in the system's output that was not seen in its input incurs a penalty of -1.

During decoding this is implemented efficiently as a matrix of dimensions batch size $\times$ target vocabulary size where all columns that match source words are assigned 0 values, all other words -1. This feature can then be used as if it was another ensemble model and tuned with the same procedure as described above.

PEP introduces a precision-like bias into the decoding process and is a simple means to enforce a certain faithfulness with regard to the input via string matching. This is not easily accomplished within the encoder-decoder framework which abstracts away from any string representations. A recall-like variant (penalize for missing input words in the output) cannot be realized at decode-time as it is not known which words have been omitted until the very end of the decoding process. This could only work as a final re-ranking criterion, which we did not explore in this paper. The bag-of-words approach grants the NMT model the greatest freedom with regard to reordering and fluency for which these models seem to be naturally well-suited.

As before, we tune the combination on the development set. The resulting system (mt$\rightarrow$pe$\times$4 / src$\rightarrow$pe$\times$4 / pep) can again improve post-editing quality. We see a total improvement of -3.7% TER and +6.0% BLEU over the given MT baseline on the development set. The log-linear combination of different features improves over the purely monolingual ensemble by -1.8% TER and +2.1% BLEU.

6 Final results and conclusions

We submitted the output of the last system (mt$\rightarrow$pe$\times$4 / src$\rightarrow$pe$\times$4 / pep) as our final proposition for the APE shared task, and mt$\rightarrow$pe$\times$4 as a contrastive system. Table 3 contains the results on the unseen test set for our two systems (in bold) and the best system of any other submitting team as reported by the task organizers (for more details and manually judged results — which were not yet available at the time of writing — see the shared task overview paper). Results are sorted by TER from best to worse. For our best system, we see improvements of -3.2% TER and +5.5% BLEU over the unprocessed baseline 1 (uncorrected MT), and -1.5% TER and +1.5% BLEU over our contrastive system. The organizers also provide results for a standard phrase-based Moses set-up (baseline 2) that can hardly beat baseline 1 (-0.1% TER, +1.4% BLEU). Both our systems outperform the next-best submission by large margins. In the light of these last results, our system seems to be quite successful.

System	TER	BLEU
mt$\rightarrow$pe$\times$4 / src$\rightarrow$pe$\times$4 / pep	**21.52**	**67.65**
mt$\rightarrow$pe$\times$4 (contrastive)	**23.06**	**66.09**
FBK	23.92	64.75
USAAR	24.14	64.10
CUNI	24.31	63.32
Standard Moses (baseline 2)	24.64	63.47
Uncorrected MT (baseline 1)	24.76	62.11
DCU	26.79	58.60
JUSAAR	26.92	59.44

Table 3: Results on unseen test set in comparison to other shared task submissions as reported by the task organizers. For submissions by other teams we include only their best result.

We could demonstrate the following:

- Neural machine translation models can be successfully applied to APE;
- Artificial APE triplets help against early overfitting and make it possible to overcome the problem of too little training data;
- Log-linear combinations of neural machine translation models with different input languages can be used as a method of combining MT-output and source data for APE to positive effects;
- Task specific features can be easily integrated into the log-linear models and can control the faithfulness of the APE results.

Future work should include the investigation of integrated multi-source approaches like (Zoph and Knight, 2016) and better schemes of dealing with overfitting. We also plan to apply our methods to the data of last year's APE task.

7 Acknowledgements

This work is partially funded by the National Science Centre, Poland (Grant No. 2014/15/N/ST6/02330).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, CA.

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, Xiamen, China.

Hanna Béchara, Raphaël Rubino, Yifan He, Yanjun Ma, and Josef van Genabith. 2012. An evaluation of statistical post-editing systems applied to RBMT and SMT systems. In *Proceedings of COLING 2012*, pages 215–230, Mumbai, India.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Rajen Chatterjee, Marco Turchi, and Matteo Negri. 2015a. The FBK participation in the WMT15 automatic post-editing shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 210–215, Lisbon, Portugal. Association for Computational Linguistics.

Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. 2015b. Exploring the planet of the APEs: a comparative study of state-of-the-art methods for MT automatic post-editing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 156–161, Beijing, China. Association for Computational Linguistics.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 428–436, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, (2):23–38.

Ulrich Germann. 2015. Sampling phrase tables for the Moses statistical machine translation system. *Prague Bulletin of Mathematical Linguistics*, (1):39–50.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

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*, pages 177–180. Association for Computational Linguistics.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 220–224, Stroudsburg, PA, USA. Association for Computational Linguistics.

Santanu Pal, Mihaela Vela, Sudip Kumar Naskar, and Josef van Genabith. 2015. USAAR-SAPE: An English–Spanish statistical automatic post-editing system. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 216–221, Lisbon, Portugal. Association for Computational Linguistics.

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*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

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

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical phrase-based post-editing. In *Proceedings of the Conference of the North American*

Chapter of the Association for Computational Linguistics, pages 508–515, Rochester, New York. Association for Computational Linguistics.

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*, pages 223–231, Cambridge, Massachusetts.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: 28th Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112, Montreal, Canada.

Guillaume Wisniewski, Nicolas Pécheux, and François Yvon. 2015. Why predicting post-edition is so hard? failure analysis of LIMSI submission to the APE shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 222–227, Lisbon, Portugal. Association for Computational Linguistics.

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

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *arXiv preprint arXiv:1601.00710*.