Noising Scheme for Data Augmentation in Automatic Post-Editing

WonKee Lee[1], **Jaehun Shin**[1],
Baikjin Jung[1], **Jihyung Lee**[1], **Jong-Hyeok Lee**[1,2]
[1]Departmet of Computer Science and Engineering,
[2]Graduate School of Artificial Intelligence,
Pohang University of Science and Technology (POSTECH), Republic of Korea
{wklee, jaehun.shin, bjjung, jihyung.lee, jhlee}@postech.ac.kr

Abstract

This paper describes POSTECH's submission to WMT20 for the shared task on Automatic Post-Editing (APE). Our focus is on increasing the quantity of available APE data to overcome the shortage of human-crafted training data. In our experiment, we implemented a noising module that simulates four types of post-editing errors, and we introduced this module into a Transformer-based multi-source APE model. Our noising module implants errors into texts on the target side of parallel corpora during the training phase to make synthetic MT outputs, increasing the entire number of training samples. We also generated additional training data using the parallel corpora and NMT model that were released for the Quality Estimation task, and we used these data to train our APE model. Experimental results on the WMT20 English-German APE data set show improvements over the baseline in terms of both the TER and BLEU scores: our primary submission achieved an improvement of -3.15 TER and +4.01 BLEU, and our contrastive submission achieved an improvement of -3.34 TER and +4.30 BLEU.

1 Introduction

There has been a surge of interest in developing Automatic Post-Editing (APE) models, which is capable of automatically correcting errors produced by a machine-translation (MT) system, and thus is an attractive way to improve the quality of the MT output. Currently, sequence-to-sequence modeling has become a dominant approach to constructing APE models (Chatterjee et al., 2019, 2018), which requires a large quantity of training samples. However, APE data[1] — comprising triplets of three texts: source (*src*), a machine-translation (*mt*) of *src*, and a human-crafted post-edited sentence (*pe*) of *mt* — is too small and costly to acquire. Consequently, the lack of APE data becomes a great obstacle to a satisfactory performance of sequence-to-sequence models.

To reduce such data scarcity, there have been several attempts at constructing synthetic APE data (Negri et al., 2018; Junczys-Dowmunt and Grundkiewicz, 2016). Most notably, Negri et al. (2018) proposed a simple but effective way to construct a large-scale synthetic APE data set eSCAPE (*src*, *mt*, *ref*), of which *src* and *ref* is the source and target text of freely available parallel corpora, respectively, and *mt* is a translation of *src* produced by the MT system that had been trained on those parallel corpora.

As eSCAPE has shown to be beneficial in training APE models (Chatterjee et al., 2019), it has become feasible to train deep APE models and also what most recent works have been relying on so far. Nevertheless, the availability of a limited quantity of parallel corpora may not only be insufficient, but also vary depending on the language pair, that is, while some language pairs have plenty of resources, some others have relatively few resources. We thus argue that further works to supply additional resources should be needed to mitigate the potential data scarcity.

In this work, we introduce a noising scheme by which corrupted texts (ref_{noise}) are produced from *ref* of parallel corpora, resulting in additional APE triplets (src, ref_{noise}, ref), where *src* and *ref* is the source and target text of parallel corpora, respectively. During post-editing, certain editing operations including the word insertion, deletion, substitution, and shifting are applied to translated texts (noisy texts) for error correction. Thus, we applied such operations to target texts (clean texts) of parallel corpora to inject errors in reverse. Moreover, to simulate the quantity of errors that the target MT system produces, we refer to the distribution of "Translation Error Rate" (TER) (Snover et al., 2006) occurring in the actual APE data to determine the quantity of errors to be injected.

[1]http://www.statmt.org/wmt20/ape-task.html

Proceedings of the 5th Conference on Machine Translation (WMT), pages 783–788
Online, November 19–20, 2020.

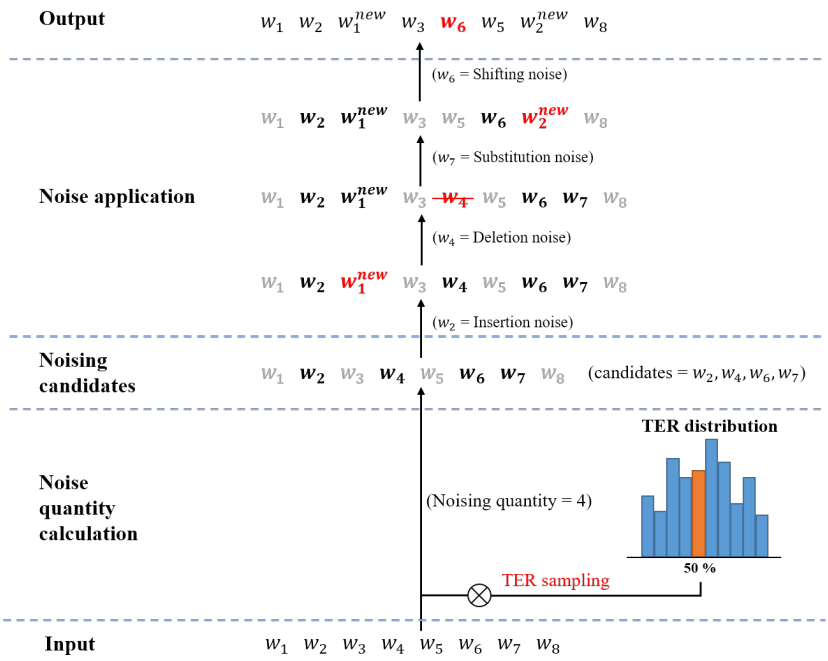

Figure 1: An illustration of the noising procedure

While we trained our models using the noising module, we supplied to the models synthetic APE data in addition to the WMT'20 APE data set as training data. The synthetic data was produced by using the eSCAPE method, which uses parallel corpora and a trained NMT model. We observed that models with noising module improved up to about -0.7 TER and +1.45 BLEU on the English-German (EN-DE) WMT'20 APE validation set compared to models without noising. Finally, our primary and contrastive submission to the WMT'20 APE shared task respectively recorded 28.41 TER and 54.22 BLEU, and 28.22 TER and 54.51 BLEU on the blind EN-DE WMT'20 APE test set.

2 Related Work

Noise injection to input sentences has become a popular method to let auto-encoders (Hill et al., 2016; Vincent et al., 2008) or pre-trained language models (Lewis et al., 2019; Devlin et al., 2019) learn how to reconstruct the original input. Because post-editing is a process of reconstructing corrupted translations, simulating corrupted MT outputs by injecting noise to the target sentence is a way to get synthetic APE training samples.

In the APE task, Xu et al. (2019) employed a data noising technique that incorporates a noise vector generated from a Gaussian or uniform distribution into the word embedding vector. However, their noising process has an effect on all tokens in a sequence, whereas only certain tokens in a given MT output are to be corrected in the APE process.

3 Method

3.1 Post-Editing Noise

Post-editing of *mt* texts requires four editing operations: insertion, deletion, substitution, and shifting. In other words, *mt* texts contain the following types of errors (The examples on the left side and right side are *mt* and *pe*, respectively.):

- Insertion operation implies that *mt* includes **deletion** errors:
  `We _ the world` → `We are the world`
- Deletion operation implies that *mt* includes **insertion** errors:
  `We are in the world` → `we are the world`
- Substitution operation implies that *mt* includes **substitution** errors:
  `We is the world` → `we are the world`
- Shifting operation implies that *mt* includes **shifting** errors:
  `We the world are` → `we are the world`

Considering the characteristics of editing operations, applying these operations to a clean text can simulate a corrupted *mt* text that can be post-edited to the original text. Thus, we corrupted a portion of words in a target text of parallel corpora, yielding a

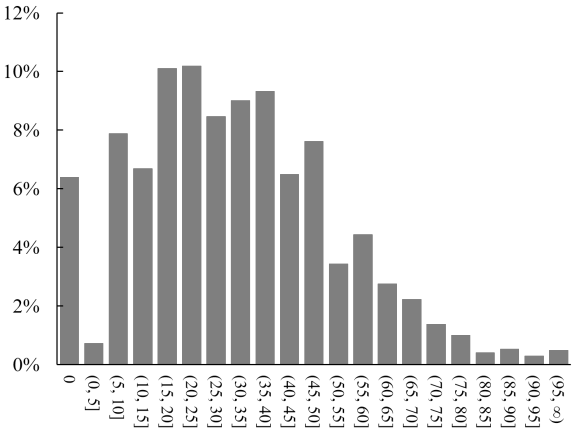

Figure 2: The categorical TER distribution of the WMT'20 training data, representing the proportion y [%] of samples belong to a specific TER range x.

new synthetic *mt* text which form a new synthetic triplet together with the corresponding source and target text in the parallel corpora.

3.2 Noising Procedure

Given an input sentence, we consider a noising procedure (Figure 1) (1) that specifies the quantity of words to become noise; (2) that selects specific words that will become noise according to the specified quantity; (3) and that determines the types of noise to be injected into the selected words.

Noising Quantity and Candidate Selection

The first step is to specify the quantity of words that will become noise in a given input sentence. Choosing a static proportion can be one option (Devlin et al., 2019), but imitating the proportion of errors that the target MT system produces would be more helpful to simulate the original *mt* text, considering that APE aims to correct the output produced by a particular MT system.

Accordingly, we refer to TER scores between *mt* and *pe* of the WMT'20 train set, which indicate the proportion of errors in *mt* that need to be corrected. Specifically, a TER range (e.g. $(45, 50]$) is drawn from the TER distribution in intervals of 5 (Figure 2), and then a specific value (e.g. 48) uniformly sampled from that range will be used as the error rate (e.g. $48 \rightarrow 0.48$). Finally, the noising quantity is calculated by multiplying the error rate by the input length. After specifying the noising quantity, we randomly select noising candidates among words in a given sentence according to the specified quantity.

Noise Application

Once the noising candidates have been selected, we now need to determine the types of noise that will be assigned to each candidate, and this process relies entirely on randomness. In particular, we produce four random numbers, making their summation equal to the noising quantity, and then this numbers are used as the quantity for each of the 'insertion', 'deletion', 'substitution', and 'shifting' post-editing noise. According to the quantity of each noising type, each type of noise is applied to words that are randomly selected among the noising candidates. Here, we present an example scenario as follows:

- Suppose that the noising quantity is 5 and the noising candidates are $w_1, w_4, w_7, w_9, w_{11}$ where w_i represents an input word in the i-th position.
- Given that the randomly selected numbers are $\{1, 2, 0, 2\}$, according to each of these four selected numbers, the words are randomly selected among the noising candidates, forming four subsets of selected words. (e.g.: $\{\{w_4\}, \{w_1, w_9\}, \{\emptyset\}, \{w_7, w_{11}\}\}$).
- Finally, one corresponding noise operation is applied to each subset of selected words. e.g.: $\{w_4\}$: insertion noise, $\{w_1, w_9\}$: deletion noise, $\{\emptyset\}$: substitution noise, $\{w_7, w_{11}\}$: shifting noise.

4 Experiment

4.1 Setup

Data. We collected publicly available parallel corpora that are listed on the WMT'20 Quality Estimation (QE) task webpage[2], which had been used to train the MT system by which *mt* texts of the WMT'20 QE corpus had been produced, and then we generated about 20M synthetic APE triplets (*src*, *mt*, *ref*) in the same manner as the eSCAPE method by using the pretrained MT system[3] that has been released on the QE task webpage. This synthetic APE triplets were used together with the official APE train set to train our APE model and a BPE model[4] by which we obtained a shared vocabulary of 32K subwords. During the training phase, the collected parallel corpora were used to

[2]http://www.statmt.org/wmt20/quality-estimation-task.html

[3]https://github.com/facebookresearch/mlqe

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

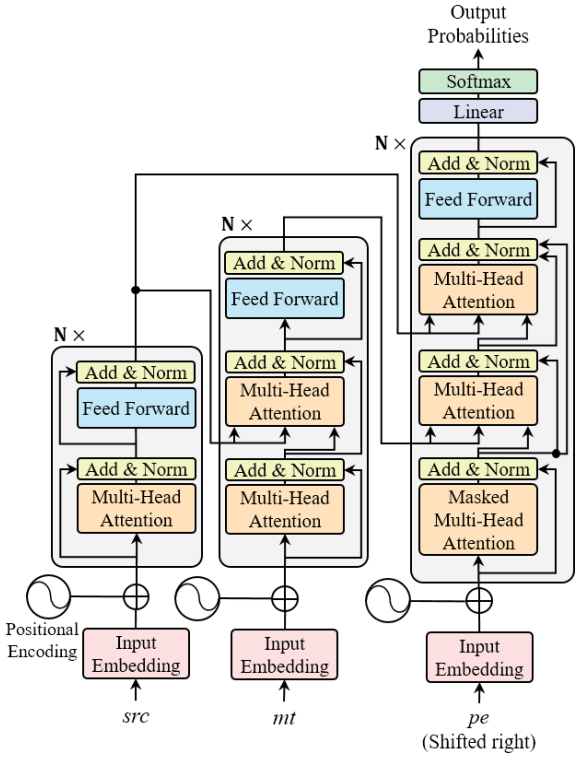

Figure 3: The structure of our APE model.

generate another synthetic APE triplets produced by our noising approach.

Model configuration. We adopted the "sequential APE model" proposed by Lee et al. (2019) to construct our APE model (Figure 3) by applying some minor modifications: the ReLU activation (Nair and Hinton, 2010) in the 'feed-forward' layers was replaced with the GELU activation (Hendrycks and Gimpel, 2016), an additional residual connection between the outputs of the first and third multi-head attention layer was added in the decoder. Additionally, we removed some details such as "stack-level attention" and "future masking to *mt*". We set our model's hyperparameters as follows: the size of the word embedding and all hidden dimensions at 768; the size of the inner dimension of feed-forward layers at 3072; 8 heads; 6 layers; and dropout rate at 0.1. The model was optimized using Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9, \beta_2 = 0.998$, and $\epsilon = 1e^{-8}$, and we used the same learning rate scheduling as Vaswani et al. (2017) with 15,000 warmup steps. We implemented and trained our model by using the OpenNMT-py[5] framework.

[5]https://github.com/OpenNMT/OpenNMT-py

Model	WMT'20 dev		WMT'20 test	
	TER	BLEU	TER	BLEU
Baseline	31.37	50.37	31.56	50.21
APE_{base}	29.42	52.32	–	–
+Noise	28.72	53.77	–	–
Submission (Ensemble)				
Primary	28.70	53.77	28.41	54.22
Contrastive	–	–	28.22	54.51

Table 1: The evaluation results. APE_{base} stands for the model trained without using our noising approach.

4.2 Training Details

Training procedure. Training of the APE model is composed of two steps. At the first step, both the synthetic triplets and WMT'20 training data are used to train the model. Every time each training batch is assigned to the model, 25% of the synthetic triplets (*src*, *mt*, *ref*) in the batch are replaced with another synthetic triplets (*src*, ref_{noise}, *ref*) by applying the noising procedure (§3.2) to each *ref*. We here set the batch size at 33,000 tokens. The following step is to fine-tune the model by only using the WMT'20 data, starting at the convergence point found in the first step. At this step, we set the batch size at 1,024 tokens.

Ensemble. We trained two ensemble models for submission. Our primary submission (TERNoise-Ops-Ens8) is an ensemble of eight runs. We first selected the top five runs, which had the lowest TER on the development set, for three individual weight initializations. To form the ensemble model, we then selected among them the top two runs, for each of four edit operations, that make corrections most frequently. Our contrastive submission (TERNoise-nFold-Ens8) is also an ensemble of eight runs. Aiming for generalization to unseen data, all runs were trained and validated in a 4-fold setting on a data set into which the training data and development data had been merged. Then we selected the top two runs for each fold to form the ensemble model.

4.3 Result

Table 1 presents our evaluation results. As the WMT20 test data is blind to users, we were not able to conduct an evaluation on the test data set, but we observed that applying our noising scheme improved the post-editing quality of the model on the development data set. As a result, our primary submission, which is an ensemble of models adopt-

ing the noising scheme, showed an improvement of -3.15 TER score and $+4.01$ BLEU score on the test data set. In addition, our contrastive submission, which is an ensemble of models trained on the separate training data set to seek generality performance on unseen data, showed better results than our primary submission, resulting in its high generalization capability to the unseen the WMT20 test data.

5 Conclusion

We propose a noising scheme to supply APE models with synthetic APE triplets during the training phase. Our noising scheme is designed based on the error types that are defined in the APE task, and the quantity of noise that are injected during the training phase are determined in consideration of the distribution of those error types in the official training data. According to the experimental results, applying our noising scheme to APE models showed an improvement of the post-editing quality in terms of both TER and the BLEU scores, which indicates that our noising scheme was effective in training APE models although there may be differences between the synthesized errors and the actual MT errors. Therefore, in future work, we will aim to reduce those gaps caused by our noising scheme.

Acknowledgments

This work was supported by Institute for Information & communications Tecnology Promotion (IITP) grant funded by the Korea government (MISP) (No. 2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH) and R7119-16-1001, Core technology development of the real-time simultaneous speech translation basd on knowledge enhancement), and was results of a study on the "HPC Support" Project, supported by the 'Ministry of Science and ICT" and NIPA.

References

Rajen Chatterjee, Christian Federmann, Matteo Negri, and Marco Turchi. 2019. Findings of the WMT 2019 shared task on automatic post-editing. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 11–28, Florence, Italy. Association for Computational Linguistics.

Rajen Chatterjee, Matteo Negri, Raphael Rubino, and Marco Turchi. 2018. Findings of the WMT 2018 shared task on automatic post-editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 710–725, Belgium, Brussels. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377.

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: Volume 2, Shared Task Papers*, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

WonKee Lee, Jaehun Shin, and Jong-Hyeok Lee. 2019. Transformer-based automatic post-editing model with joint encoder and multi-source attention of decoder. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 112–117.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. 2018. Escape: a large-scale synthetic corpus for automatic post-editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

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*, volume 200. Cambridge, MA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

Hongfei Xu, Qiuhui Liu, and Josef van Genabith. 2019. Uds submission for the wmt 19 automatic post-editing task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 147–152, Florence, Italy. Association for Computational Linguistics.