Automatic Post-Editing of Machine Translation: A Neural Programmer-Interpreter Approach

Thuy-Trang Vu
Faculty of Information Technology
Monash University, Austrlia
vuth0001@student.monash.edu

Gholamreza Haffari
Faculty of Information Technology
Monash University, Austrlia
first.last@monash.edu

Abstract

Automated Post-Editing (PE) is the task of automatically correcting common and repetitive errors found in machine translation (MT) output. In this paper, we present a neural programmer-interpreter approach to this task, resembling the way that humans perform post-editing using discrete edit operations, which we refer to as programs. Our model outperforms previous neural models for inducing PE programs on the WMT17 APE task for German-English up to +1 BLEU score and -0.7 TER scores.

1 Introduction

Automatic post-editing (APE) is the automated task that aims to correct common systematic and repetitive errors found in machine translation (MT) output. APE systems can also be used to adapt general-purpose MT output to specific domains without re-training MT models, or to incorporate information which is not available or expensive to compute at MT decoding stage. Post-editing is considered as the modification process of a machine translated text with a minimum labor effort rather than re-translation from scratch.

Previous studies in neural APE have primarily concentrated on formalizing APE as a monolingual MT problem in the target language, with or without conditioning on the source sentence (Pal et al., 2016; Chatterjee et al., 2017). MT approach has suffered from *over-correction* where APE system performs unnecessary correction leading to paraphasing and the degradation of the output quality (Bojar et al., 2016, 2017).

Recent works (Libovický et al., 2016; Berard et al., 2017) have attempted to learn the predict of sequence of post-editing operations, e.g. insertion and deletion, to induce APE *programs* to turn the machine translated text into the desired output. Previous program induction approaches suffer from *over-cautiousness*, where the APE system tends to keep the machine translated text without any modification (Bojar et al., 2017).

In this paper, we propose a programmer-interpreter approach to the APE task to address the over-cautiousness problem. Our architecture includes an interpreter module, which executes the previous editing action before generating the next one. This is in contrast to the previous work, where the *full* program is induced before it is *executed*. The ability of execution *immediately* at every time step provides a proper conditioning context based on the actual partial edited sentence to assist better prediction of the next operation. Moreover, the execution module can be pre-trained on monolingual target text, enabling our architecture to benefit from monolingual data in addition to PE data, which is hard to obtain. Our model is jointly trained on translation task and APE program induction task. The multi-task architecture allows the model to reconstruct the source-target alignment of the black-box MT system and inject it into post-editing task.

We compare our programmer-interpreter architecture against previous works on the English-German APE based on the data for this task in WMT16 and WMT17. Compared to the previous work on APE program induction, our architecture achieves improvements up to +1 BLEU and -0.7 TER scores. Our analysis also shows that APE programs generated by our model are not only better at correcting errors but also attempt to perform more editing actions.

2 Related Work

Pal et al. (2016) has applied the SEQ2SEQ model to APE. Their monolingual MT learned to post-edit English-Italian Google Translation output and

Source	Does not have a menu bar .
MT	Weist$_1$ keine$_2$ Menüleiste$_3$ angezeigt$_{4\ldots5}$
Reference	Weist $_1$ keine$_2$ Menüleiste$_3$ auf$_{4\ldots5}$
APE prog.	*KEEP$_1$ KEEP$_2$ KEEP$_3$ auf $_4$ DEL$_4$ KEEP$_5$ STOP*

Figure 1: An example of PE program when executing on MT would return the PE reference. The color denotes the alignment between MT, reference PE and the APE program. The number subscript shows the edit position in original MT sentence.

was able to reduce the preposition related errors. Blindly performing edition over MT output, the monolingual APE has difficulty to correct missing word or information in the source sentence. Neural multi-source MT architectures are applied to better capture the connection between the source sentence/machine translated text and the PE output (Libovický et al., 2016; Varis and Bojar, 2017; Junczys-Dowmunt and Grundkiewicz, 2017). Chatterjee et al. (2017) ensemble several different models including monolingual MT (TGT→PE), bilingual MT (SRC→PE), and multi-source (SRC,TGT→PE). Libovický et al. (2016); Berard et al. (2017) have proposed learning to predict the sequence of edit operations, aka the program, to produce the post-editing sentence (c.f. §3).

Our work is motivated by Ling et al. (2017) on learning to indirectly solve an algebraic word problem by inducing a program which generates the answer together with an explanation. It further builds up on recent work on neural programmer-interpreter (Reed and De Freitas, 2016), where a neural network programmer learns to program an interpreter. The architecture is then trained using expert action trajectories as programs.

3 The NPI-APE Approach

Given a source sentence $\boldsymbol{s}$ and a machine translated sentence $\boldsymbol{m}$, the goal is to find a post-edited sentence $\boldsymbol{t} = \arg\max_{\boldsymbol{t}'} P_{\text{ape}}(\boldsymbol{t}'|\boldsymbol{m}, \boldsymbol{s})$ where $P_{\text{ape}}(.)$ is our probabilistic APE model. In our proposed approach, we aim to find an editing action sequence $\boldsymbol{z}$ to *execute* in order to generate the desired post-edited sentence,

$$P_{\text{ape}}(\boldsymbol{t}|\boldsymbol{m}, \boldsymbol{s}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} P_{\text{ape}}(\boldsymbol{t}, \boldsymbol{z}|\boldsymbol{m}, \boldsymbol{s}).$$

We decompose the joint probability of a program and an output as:

$$P_{\text{ape}}(\boldsymbol{z}, \boldsymbol{t}|\boldsymbol{m}, \boldsymbol{s}) = \prod_{i=1}^{|\boldsymbol{z}|} P_{\text{prog}}(z_i|\boldsymbol{t}_{\leq j_{i-1}}, \boldsymbol{m}, \boldsymbol{s}) \quad (1)$$

$$\times P_{\text{intp}}(t_{j_i}|m_{k_i}, z_i) \quad (2)$$

where $P_{\text{prog}}(z_i|\boldsymbol{t}_{\leq j_{i-1}}, \boldsymbol{m}, \boldsymbol{s})$ is the programmer's probability in producing the next edit operation z_i given the post edited output $\boldsymbol{t}_{\leq j_{i-1}}$ generated from the operations so far $\boldsymbol{z}_{\leq i-1}$, and $P_{\text{intp}}(t_{j_i}|m_{k_i}, z_i)$ is the interpreter's probability of outputing t_{j_i} given the edit operation z_i and the MT word m_{k_i}.

Following Berard et al.(2017), our action sequence is performed on the MT sentence from left to right. At each position, we can take one of the following editing operations: (i) KEEP to keep the word and go to the next word, (ii) DELETE to delete the word and go to the next word, (iii) INSERT(WORD) to insert a new WORD and stay in that position, or (iv) STOP to terminate the process. In other words, the size of the operation set equals the size of the target vocabulary plus three, where we add the symbols KEEP, DELETE, and STOP as new tokens. Furthermore, j_i is the number of KEEP and INSERT(WORD) operations, and k_i is the number of KEEP and DELETE operations in the sequence of operations $\boldsymbol{z}_{\leq i}$. This *hard attention* mechanism is the outcome of the semantics of the operations, and injects task knowledge into the model. Moreover, $P_{\text{intp}}(t|m, z)$ is 1 if the output word t is consistent with performing the operation z on m, and zero otherwise.

Our decomposition of the joint probability of a program and post-edited output is distinguished from that proposed in (Berard et al., 2017), $P_{\text{ape}}(\boldsymbol{t}, \boldsymbol{z}|\boldsymbol{m}, \boldsymbol{s}) = P_{\text{intp}}(\boldsymbol{t}|\boldsymbol{z}, \boldsymbol{m}) P_{\text{prog}}(\boldsymbol{z}|\boldsymbol{m}, \boldsymbol{s})$. Crucially, in our decomposition (eqns 1 and 2), the programming and interpreting are interleaved at each position, whereas in (Berard et al., 2017) the programming is fully done before the interpretation phase and they are independent.

3.1 Neural Architecture and Joint Training

The architecture consists of three components (i) A SEQ2SEQ model to translate the source sentence to the target in the forced-decoding mode (MT), (ii) A SEQ2SEQ model to incrementally generate the sequence of edit operations (Action Generator), and (iii) An RNN to summarize the post edited sequence of words produced from the execution of actions generated so far (Interpreter).

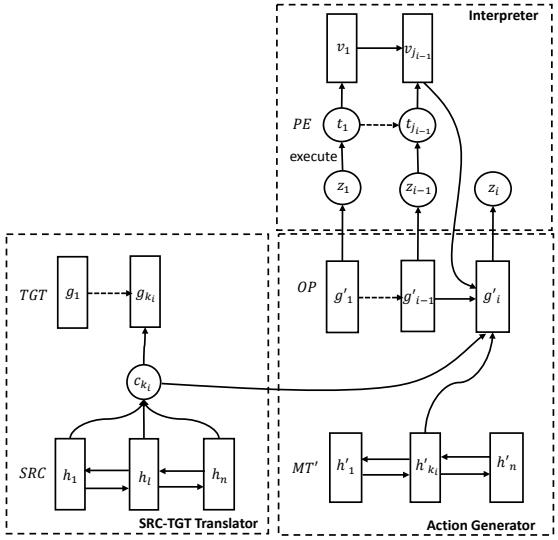

Figure 2: Our proposed NPI-APE model

The encoder and decoder of the MT component are a bidirectional and unidirectional LSTM, whose states are denoted by $\boldsymbol{h}_l$ and $\boldsymbol{g}_k$, respectively. Similarly, the encoder and decoder of the AG (Action Generator) component are a bidirectional and unidirectional LSTM, whose states are denoted by $\boldsymbol{h}'_k$ and $\boldsymbol{g}'_i$, respectively. The states of the unidirectional RNN in the interpreter are denoted by $\boldsymbol{v}_j$.

The next edit operation z_i is generated from the decoder state of the AG $\boldsymbol{g}'_i$ (see Figure 2), which is computed from the previous state $\boldsymbol{g}'_{i-1}$ and a context including the following: (i) $\boldsymbol{h}'_{k_i}$ where k_i is the index of the MT output word currently processed, (ii) $\boldsymbol{c}_{k_i}$ which is the context vector from the MT component when generating the current target word, and (iii) $\boldsymbol{v}_{j_{i-1}}$ which is the last hidden state of the interpreter RNN encoding the post edited sentence generated so far.

The model is trained *jointly* for the translation task (SRC→TGT), and for the post editing task (SRC,TGT→OP,PE). For the training data, we compute the lowest-cost sequence of editing operations (OP) using dynamic programming, where the cost of insertion and deletion are 1.

4 Experiments

Dataset. We evaluate the proposed approach on the English-to-German (En-De) post-editing task in the IT domain using the data from WMT16[1] and WMT17.[2] The official WMT'16 and WMT'17 dataset contains 12K and 11K post-editing triplets (English, translated German, post-edited German) respectively in IT domain. We concatenated them to an 23K triplets. A synthetic corpus of 500K triplets (Junczys-Dowmunt and Grundkiewicz, 2016) is also available as additional training data. We performed our experiment in two different settings with and without synthetic data for comparison with Berard et al. (2017).

The RNN in the interpreter component can be thought of as a language model. This paves the way to pre-train it using monolingual text. We collect in-domain IT text from OPUS[3] from the following sections: GNOME, KDE, KDEdoc, OpenOffice, OpenOffice3, PHP and Ubuntu. After tokenizing, filtering out sentences containing special characters, and removing duplications, we obtain around 170K sentences.

Setup. There are three components in our architecture: machine translation (MT), action generator (AG), and interpreter (LM). We compare our MT+AG+LM architecture against MT+AG[4] Berard et al. (2017) which does not have the LM component. The size of the hidden dimensions (LSTMs in the MT and AG, and simple RNN in the LM component) as well as word embedding in these models is set to 128.

Furthermore, we compare against monolingual SEQ2SEQ (TGT→PE) as well as the multi-source SEQ2SEQ (SRC+TGT→PE) (Varis and Bojar, 2017). Monolingual SEQ2SEQ (TGT→PE) model is an attentional SEQ2SEQ model (Bahdanau et al., 2015) that takes target sentence as input and outputs desired PE sentence. In multi-source SEQ2SEQ (SRC+TGT→PE), we use two encoders for source and target sentences and concatenate their context vectors. In both models, the encoder and decoder contain a single layer of bidirectional and unidirectional LSTM respectively. The size of the LSTM hidden dimensions and word embedding in these models is set to 256 and 128, respectively. This ensures almost the same number of parameters (∼13M) in all architectures.

Training. We use a multi-task scenario to jointly train the parameters of the components in MT+AG as well as MT+AG+LM models. For the latter, we warm start the embedding of the target words with

[1] http://www.statmt.org/wmt16/ape-task.html
[2] http://www.statmt.org/wmt17/ape-task.html
[3] http://opus.nlpl.eu/
[4] The AG decoder in MT+AG conditions a state on the last generated action as well.

	Model	dev TER	dev BLEU	test2016 TER	test2016 BLEU	test2017 TER	test2017 BLEU
	Original MT	24.81	62.92	24.76	62.11	24.48	62.49
12K	TGT → PE	63.76	21.32	60.96	22.11	65.13	18.13
	SRC+TGT → PE	51.41	34.04	48.27	35.24	50.98	31.52
	MT+AG	23.74	65.95	23.53	65.22	23.77	64.34
	MT+AG+LM	**23.36**†	**66.24**	**23.24**†	**65.53**†	**23.45**†	**64.65**†
500K+12K	TGT → PE	50.91	30.88	48.62	32.55	52.07	27.98
	SRC+TGT → PE	30.97	53.97	30.20	53.92	32.82	50.30
	MT+AG	22.82	66.51	22.87	65.67	23.58	64.35
	MT+AG+LM	**22.67**	**67.17**†	**22.53**†	**66.30**†	**23.03**†	**65.31**†
23K	TGT → PE	57.02	27.87	55.52	27.8	60.06	22.78
	SRC+TGT → PE	38.06	47.42	36.61	47.93	39.86	43.52
	MT+AG	23.10	66.60	22.82	66.15	23.14	65.19
	MT+AG+LM	**22.61**†	**67.19**†	**22.42**†	**66.53**†	**22.84**†	**65.52***†
500K+23K	TGT → PE	48.89	34.29	47.12	34.75	51.13	29.59
	SRC+TGT → PE	28.61	57.47	27.79	57.61	30.34	53.26
	MT+AG	22.38	67.34	22.14	66.53	22.71	65.34
	MT+AG+LM	**21.99***†	**67.50***	**22.07***	**66.67***	**22.58***	**65.50**

Table 1: TER and BLEU scores of our model (MT+AG+LM) v.s. the rest on various data conditions for the EN-DE post-editing task. **bold**: Best results within a data condition; *: Best results across data conditions; †: Statistically significant compared to MT+AG with pvalue ≤ 0.05.

those obtained from training the LM component on the monolingual text[5].

All models are trained with SGD, where the learning rate is initialised to 1 and decays after each epoch. The learning rate is decayed 0.8 after every epoch for model trained with official post-editing data, and 0.5 every half epoch for model with synthetic data. All models use the same vocabulary on the same data condition. The Vocabulary size is 30K for large dataset experiments, and 27K/19K for the 23K/12K data conditions. In all experiments, the best model is selected based on TER on the validation set. For decoding, we use beam search with the beam size of 10.

4.1 Results

Table 1 shows the result on different training datasets to compare our model against the baselines. Original MT is the strong standard *do-nothing* baseline, i.e. copying the MT translation as the PE output. In all settings, our MT+AG+LM models outperforms the MT+AG and monolingual/multi-source SEQ2SEQ models. Specifically, our model outperform MT+AG in 500K+12K training condition by almost 1 BLEU score on test2017.

As expected, the models trained on 23K data perform better than those trained on 12K; further gains are obtained by adding 500K synthetic data.

[5]The MT, AG, and LM components share the target word embedding table in our model. Similarly, MT and AG components share the target embedding in the MT+AG model.

	Model	Sentences Mod	Sentences Prec.	Actions Mod.	Actions Prec.
dev	MT+AG_23K	546	61.17%	986	60.45%
	MT+AG+LM_23K	558	65.59%	1098	65.21%
	MT+AG_500+23K	673	61.81%	1477	62.02%
	MT+AG+LM_500+23K	682	65.84%	1560	68.21%
test2016	MT+AG_23K	1039	60.92%	1766	62.63%
	MT+AG+LM_23K	1041	66.09%	1919	65.61%
	MT+AG_500+23K	1269	63.04%	2728	63.71%
	MT+AG+LM_500+23K	1251	63.87%	2814	63.50%
test2017	MT+AG_23K	897	55.85%	1535	57.79%
	MT+AG+LM_23K	952	61.24%	1853	60.44%
	MT+AG_500+23K	1182	54.99%	2599	55.71%
	MT+AG+LM_500+23K	1180	56.78%	2653	59.67%

Table 2: Sentence precision and action precision of models trained on 23K and 500K+23K dataset.

Interestingly, training MT+AG and MT+AG+LM models on 23K data lead to better TER/BLEU than those trained on 500K+12K. This implies the importance of in-domain training data, as the synthetic corpus is created using general domain *Common-Crawl* corpus.

4.2 Analysis

We perform fine-grained analysis of the changes made by our model vs MT+AG in order to understand the sources of improvements. For different data conditions, Table 2 shows the number of modified sentences by each model as well as the sentence level precision defined as the fraction of sentences with improved TER. Moreover, it reports the total number of actions generated by the model on sentences with improved TER, as well as the precision of such actions, i.e. the fraction of those observed in the ground truth action trajecto-

ries.

As reported in Berard et al. (2017), one major challenge of predicting action sequences is the class imbalance. The model is often too conservative about its edits, and tends to use the KEEP far more than the INSERT and DELETE actions. Our Programmer-Interpreter model tackles this problem, as evidenced by its comparable number of modified sentences, but with higher sentence and action level precision in almost all cases.

5 Conclusion

In this paper, we have presented a neural programmer-interpreter approach to automated post-editing of MT output. Our approach interleaves generating the sequence of edit actions by a programmer component, and executing those actions with an interpreter component. This leads to better capturing the history of the past generated actions when generating the next action. Our approach achieves up to +1 BLEU and -.7 TER improvement compared to a variant in which programming is not interleaved with execution. Future work includes inducing macro-actions composed of simpler building block actions.

Acknowledgments

The authors are grateful to the anonymous reviewers for their helpful comments and corrections. This work was supported by the Multi-modal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) (www. massive.org.au), and partially supported by a Google Faculty Award to GH and the Australian Research Council through DP160102686.

References

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

Alexandre Berard, Laurent Besacier, and Olivier Pietquin. 2017. LIG-CRIStAL submission for the WMT 2017 automatic post-editing task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 623–629, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

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*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Rajen Chatterjee, M. Amin Farajian, Matteo Negri, Marco Turchi, Ankit Srivastava, and Santanu Pal. 2017. Multi-source neural automatic post-editing: FBK's participation in the WMT 2017 APE shared task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 630–638, Copenhagen, Denmark. Association for Computational Linguistics.

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, Berlin, Germany. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2017. An exploration of neural sequence-to-sequence architectures for automatic post-editing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 120–129. Asian Federation of Natural Language Processing.

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*, pages 646–654, Berlin, Germany. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

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*, volume 2, pages 281–286.

Scott Reed and Nando De Freitas. 2016. Neural programmer-interpreters. In *Proceedings of International Conference on Learning Representations (ICLR) 2016*.

Dusan Varis and Ondřej Bojar. 2017. CUNI system for WMT17 automatic post-editing task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 661–666, Copenhagen, Denmark. Association for Computational Linguistics.