ISSN 2499-4553

IJCoL

Italian Journal of Computational Linguistics

Rivista Italiana di Linguistica Computazionale

Volume 5, Number 1
june 2019

aA Accademia university press

Registrazione presso il Tribunale di Trento n. 14/16 del 6 luglio 2016

Rivista Semestrale dell'Associazione Italiana di Linguistica Computazionale (AILC)

Associazione Italiana di Linguistica Computazionale

direttore responsabile
Michele Arnese

isbn 978-88-31978-89-7

Accademia University Press
via Carlo Alberto 55
I-10123 Torino
info@aAccademia.it
www.aAccademia.it/IJCoL_5_1

aAccademia university press

IJCoL

Volume 5, Number 1
june 2019

CONTENTS

Multi-source Transformer for Automatic Post-Editing of Machine Translation Output

Amirhossein Tebbifakhr*
Fondazione Bruno Kessler
Università di Trento

Matteo Negri**
Fondazione Bruno Kessler

Marco Turchi†
Fondazione Bruno Kessler

Automatic post-editing (APE) of machine translation (MT) is the task of automatically fixing errors in a machine-translated text by learning from human corrections. Recent APE approaches have shown that best results are obtained by neural multi-source models that correct the raw MT output by also considering information from the corresponding source sentence. In this paper, we pursue this objective by exploiting Transformer (Vaswani et al. 2017), the state-of-the-art architecture in MT. Our approach presents several advantages over previous APE solutions, both from the performance perspective and from an industrial deployment standpoint. Indeed, besides competitive results, our Transformer-based architecture is faster to train (thanks to parallelization) and easier to maintain (thanks to the reliance on a single model rather than a complex, multi-component architecture). These advantages make our approach particularly appealing for the industrial sector, where scalability and cost-efficiency are important factors, complementary to pure performance. Besides introducing a novel architecture, we also validate the common assumption that training neural APE systems with more data always results in stronger models. Along this direction, we show that this assumption does not always hold, and that fine-tuning the system only on small in-domain data can yield higher performance. Furthermore, we try different strategies to better exploit the in-domain data. In particular, we adapt reinforcement learning (RL) techniques to optimize our models by considering task-specific metrics (i.e. BLEU and TER) in addition to maximum likelihood. Our experiments show that, alone, the multi-source approach achieves slight improvements over a competitive APE system based on a recurrent neural network architecture. Further gains are obtained by the full-fledged system, fine-tuned on in-domain data and enhanced with RL optimization techniques. Our best results (with a single multi-source model) significantly improve the performance of the best (and much more complex) system submitted to the WMT 2017 APE shared task.

1. Introduction

Recent advances in Machine Translation (MT) have made the post-editing of raw MT output more cost-efficient in industry settings compared to human translation from scratch. In this *translation as post-editing* workflow, firstly the source text is translated

* Fondazione Bruno Kessler, Via Somarive 18, 38123 Povo (Trento), Italy. E-mail: atebbifakhr@fbk.eu
** Fondazione Bruno Kessler, Via Somarive 18, 38123 Povo (Trento), Italy. E-mail: negri@fbk.eu
† Fondazione Bruno Kessler, Via Somarive 18, 38123 Povo (Trento), Italy. E-mail: turchi@fbk.eu

by means of an MT system. Then, the machine-translated text is revised by a human expert in order to correct possible errors. Although the reliance on high-quality MT systems reduces the amount of effort needed by professional translators, state-of-the-art MT output is not yet perfect: systematic errors may still occur, which require repeated human corrections of similar mistakes. Automatic Post-Editing (APE) aims to automatize this process and, in doing so, to speed-up, reduce the workload and the overall costs of professional translation.

The APE task can be cast as a "monolingual translation" problem (i.e. the translation from raw MT output into publishable material), in which a model is trained, in a supervised fashion, on datasets comprising (*source-text*, *MT-output*, *human_post-edit*) triplets. Cast in this way, the problem can be approached thanks to the wealth of data daily produced by modern industry workflows based on the translation as post-editing paradigm. In terms of approaches, APE research followed a similar path to that of MT. Early approaches based on the statistical paradigm (Simard et al. 2007) were recently replaced by more advanced and effective neural solutions. Among them, the most effective ones (Chatterjee et al. 2015; Pal et al. 2016) operate by looking not only at the MT output to be corrected but also at the corresponding source text (i.e. correction actions are conditioned to both the texts) in order to leverage more information, resolve potential ambiguities, and eventually return more reliable corrections. However, a drawback of these state-of-the-art APE solutions is the reliance on pipelined architectures (Bojar et al. 2017), whose complexity raises training/maintenance issues and eventually reduces their usability. Indeed, current top systems typically rely on ensembling multiple recurrent neural networks (RNNs) and performing a final re-ranking step (Chatterjee et al. 2017) to select the most promising correction hypothesis. Though competitive, such architectures require training and maintaining multiple components, involving costs that reduce their appeal from the industry perspective.

To overcome these issues, in this paper we first propose a single multi-source APE system based on Transformer (Vaswani et al. 2017), the state-of-the-art architecture in MT. Our experiments show that this system can reach state-of-the-art performance on the benchmark released for the WMT 2017 APE shared task (Bojar et al. 2017). Beside the better performance, our system has two advantages compared to previous approaches. First, it is faster to train compared to previous RNN-based solutions. This is due to the fact that, in contrast to the auto-regressive nature of RNNs, in the Transformer architecture all the positions in the sentence are processed in parallel. Second, it is easier to maintain, since only one single model is trained in an end-to-end fashion, and there is no need for optimizing different components interacting with each other.

Moving a step forward in our research, we then focus on the data dimension. Indeed, although neural approaches achieve state-of-the-art results (in APE like in MT), they need to be trained on huge amounts of data. Prior work heavily built on a "the more the better" assumption (Bojar et al. 2017). Following this assumption, all the available training data are used and possibly augmented with large synthetic datasets (Junczys-Dowmunt and Grundkiewicz 2016). The way data are exploited, however, is still an unexplored problem, leaving large room for research on how to optimize their use towards better performance. Along this direction, we investigate different ways to combine gold, in-domain data with large, sub-optimal synthetic corpora. In particular, focusing on model optimization, we adapt to APE different ways to maximize the exploitation of small in-domain corpora rather than simply relying on a brute force processing of all the available data. To this aim, we choose optimization strategies that are drawn from Reinforcement learning (Ranzato et al. 2016; Shen et al. 2016). The advantage of these strategies instead of maximum likelihood is twofold. First, to make

the optimization process more reliable, they directly target task-specific, reference-based evaluation metrics (BLEU and TER) instead of maximizing the likelihood of the training data, which can have low correlation with these metrics. Second, being more directly driven by the final evaluation metrics, they are more suitable to preserve the style of the reference translations. This property is particularly important in APE, where the system should mimic the behaviour of human post-editors (i.e. perform the minimum amount of edit operations required to fix the MT errors), without "over-correcting" the machine-translated text with unnecessary complete rephrasing. Discouraging the system to drastically change the MT output (even with post-edits that are *per se* correct) is crucial to avoid penalizing the system by automatic evaluation metrics based on references built from minimal human corrections.

Our main contributions can be summarized as follows:

- We introduce a neural, multi-source APE system based on the Transformer architecture;
- We conduct a set of experiments in order to analyze the effect of using synthetic and in-domain training data to build a neural APE model;
- We explore different reinforcement learning approaches in order to maximize the usefulness of in-domain data during training;
- We evaluate our multi-source Transformer-based system and the proposed enhancements on a shared evaluation benchmark, on which we achieve state-of-the-art results.

This research extends previous work by the same authors (Tebbifakhr et al. 2018), by adding the analysis of using different type of data and different optimization strategies, which ends to state-of-the-art results in the APE task. The paper is organized as follows. In Section 2, we review the previous works on APE. In Section 3, we describe our multi-source architecture based on Transformer and introduce different model optimization strategies drawn from reinforcement learning. In Section 4, we describe our experimental setup and model configuration. In Section 5, we discuss the results of our experiments. Finally, Section 6 summarizes the paper with conclusions.

2. Related Work

The APE task was introduced for the first time by Knight and Chander (1994) to automatically insert the correct articles in Japanese to English translation. Other more recent works adopted rule-based approaches (Rosa, Mareček, and Dušek 2012), but they gained limited attention, due to involving extensive manual work in defining the rules.

A statistical approach has been proposed for the first time in (Simard et al. 2007), in which the problem is cast as a "monolingual translation" task. In particular, raw MT output is "translated" into better translations by means of a phrase-based MT (PB-SMT) system trained on (MT_output, post-edited_output) pairs. They showed that APE components trained on human corrections of machine translated texts yield substantial improvements to the original MT system output. This results in further investigation of using PBSMT systems as an APE module for different language directions, over different underlying MT system architectures, in different domains (Isabelle, Goutte, and Simard 2007; Dugast, Senellart, and Koehn 2009; Lagarda et al. 2009; Béchara et al. 2012; Rosa, Mareček, and Tamchyna 2013). In (Isabelle, Goutte, and Simard 2007) the

authors used an APE module for domain adaptation: the idea is to learn from human corrections not only how to fix MT errors, but also how to adequately translate in a specific target domain. It has shown to be useful for reducing time, effort and the overall costs of human translation in industry environments (Aziz, Castilho, and Specia 2012). Recently, the advent of deep learning (in particular neural machine translation – NMT) caused a shift from the statistical approaches to more powerful neural approaches. (Pal et al. 2016; Junczys-Dowmunt and Grundkiewicz 2016) are among the first works in APE relying on Neural Machine Translation (NMT) models. These papers rely on an encoder-decoder architecture with attention model. On different datasets, neural APE models were able to significantly improve the outputs of PBSMT systems. Building on these positive results, this paper takes a step forward by exploiting Transformer (Vaswani et al. 2017) (i.e. the state of the art architecture for MT) to approach the APE task.

Although casting the APE task as a monolingual translation gained lots of attention, it has a major drawback, which is disregarding the information about the source text. Indeed, part of the information in the source text can be lost by errors made by the MT system, which cannot be rectified by only attending on the MT output. To cope with this issue, Béchara, Ma, and van Genabith (2011) proposed a "source context-aware" variant of the work by Simard et al. (2007). This approach creates a new input sentence representation by combining each MT word with its corresponding source word. When tested with a PBSMT APE system, it results in better performance than using only the MT sentence. In neural APE, Junczys-Dowmunt and Grundkiewicz (2016) use a log-linear combination of two NMT models (source-text - post-edited_output and MT_output - post-edited_output) for translating source and raw MT output to human post-edited text, both using (Bahdanau, Cho, and Bengio 2015) architecture. In (Chatterjee et al. 2017), they show that the best results can be obtained by a single APE system that uses two different encoders to exploit the source and MT sentences in APE. In this paper, we further explore the potential of the multi-source approach by leveraging the Transformer, a more powerful NMT architecture.

More recently, addressing the problem of over-correction in APE, (Chatterjee et al. 2018) examine different strategies to combine Quality Estimation (QE) and APE. The over-correction problem happens when APE system tends to rephrase an already good translation. To prevent this problem, they propose three different approaches for integrating QE and APE: *i)* QE as an activator, which decides if the MT sentence needs to be post-edited or not, *ii)* QE as a selector, which selects the best output between the MT output and the post-edited text , and *iii)* QE as a guidance, which identifies problematic parts of the translation for post-editing.

The parameters of most NMT systems and consequently neural APE models are optimized by maximizing the likelihood of the training data. Indeed, a token-level cross-entropy loss function is defined to maximize the probability of each token in the target sequence. However, the performance of these systems is evaluated using sequence-level evaluation metrics such as BLEU (Papineni et al. 2002) and TER (Snover et al. 2006). To cope this discrepancy, different reinforcement learning methods such as REINFORCE (Ranzato et al. 2016), actor-critic (Bahdanau et al. 2016) and minimum risk training (Shen et al. 2016) have been used to directly maximize these sequence-level metrics for sequence generation tasks. All the approaches show that integrating the evaluation metric in the optimization process results in improvement in performance. These approaches that have been extensively used in NMT, have not been tested in APE. Exploring them, together with analysing their effectiveness in different data conditions

is another contribution of this paper. As a final contribution, we present state-of-the-art results obtained by our architecture on a shared evaluation benchmark.

3. A Neural APE System Based on Transformer

In this section we introduce: *i)* the Transformer architecture, *ii)* its multi-source extension and *iii)* optimization techniques, drawn from reinforcement learning, targeting the integration of task-specific metrics in the computation of the loss. Altogether, these elements represent the core of our approach to neural APE.

3.1 Transformer and its Multi-Source Extension

As discussed in the previous sections, current approaches to APE as a "monolingual translation" task rely on state-of-the-art architectures developed for neural MT. Typically, they employ deep recurrent networks (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2015) in the so-called encoder-decoder framework. In this framework, a sequence of words $[x_1, x_2, .. , x_n]$ is given to an encoder, which maps it to a sequence of continuous representations, i.e. the hidden state of the encoder. At each time step, based on these continuous representations and the generated word in the previous time step, a decoder generates the next word. This process continues until the decoder generates the end-of-sentence word. More formally, at each time step the decoder predicts the next word y_n, given the context vector c and the previously predicted words $y_{<n}$ by defining a probability over the translation $\mathbf{y}$ as follows:

$$p(\mathbf{y}) = \prod_{n=1}^{N} p(y_n|y_{<n}, c) \tag{1}$$

The context vector c is a weighted sum computed over the hidden states of the encoder. The weights used to compute the context vector are obtained by a network called attention model that finds an alignment between the target and source words (Bahdanau, Cho, and Bengio 2015). From an efficiency standpoint, a major drawback of these approaches is that, at each time step, the decoder needs the hidden state of the previous time step, thus hindering parallelization. Other approaches have been proposed to avoid this sequential dependency (e.g. using convolution as a main building block) and make parallelization possible (Gehring et al. 2017; Kalchbrenner et al. 2016). However, although they can avoid the recurrence, they are not able to properly learn the long term dependencies between words.

The Transformer architecture introduced in (Vaswani et al. 2017), set a new state-of-the-art in NMT by completely avoiding both recurrence and convolution. Since the model does not leverage word-order information, it adds positional encoding to the input word embeddings to enable capturing word order. This aspect is particularly important since, in contrast to the auto-regressive nature of RNNs, it allows Transformer to process all the input positions in parallel, with considerable savings in computation time.

In Transformer, the attention employed is a multi-headed self-attention, which is a mapping from (*query*, *key*, *value*) tuples to an output vector. The self-attention is defined

as follows:

$$\mathrm{SA}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

where Q is the query matrix, K is the key matrix and V is the value matrix, d_k is the dimensionality of the queries and keys, and SA is the computed self-attention. The multi-head attention (MH) is computed as follows:

$$\mathrm{MH}(Q,K,V) = \mathrm{Concat}\left(head_1,\ldots,head_h\right)W^O \tag{3}$$

where h is the number of attention layers (also called "heads"), $head_i$ is the self-attention computed over the i^{th} attention layer and W^O is the parameter matrix of dimension hd_v*d_{model}. The encoder layers consist of a multi-head self-attention, followed by a position-wise feed forward network. In the self-attention, the queries, keys and values matrices come from the previous layer. In the decoder, the layers have an extra encoder-decoder multi-head attention after the multi-head self-attention, where the key and value matrices come from the encoder and the query matrix comes from the previous layer in the decoder. Also, inputs to the multi-head self-attention in the decoder are masked in order to not attend to the next positions. Finally, a softmax normalization is applied to the output of the last layer in the decoder to generate a probability distribution over the target vocabulary.

In order to encode the source sentence in addition to the MT output, we employ the multi-source method (Zoph and Knight 2016), wherein the model is comprised of two separate encoders (with a different set of parameters) that respectively provide continuous representations the source sentence and the MT output. The output of the two encoders is concatenated and passed as the key in the attention. This helps for a better representation, in turn leading to more effective attention during decoding time.

3.2 Integrating Task-Specific Metrics in Loss Computation

While in the previous section we overviewed Transformer and its multi-source extension introduced in (Tebbifakhr et al. 2018), in this section we present a further extension of the approach. In particular, targeting a more effective model optimization, we discuss the application of different learning strategies based on reinforcement learning.

In general, neural MT systems are optimized to maximize the likelihood of training data (i.e. Maximum Likelihood Estimation – MLE). In order to maximize the likelihood of the given parallel corpus $\{\mathbf{x}^{(s)},\mathbf{y}^{(s)}\}_{s=1}^{S}$, they use token-level cross-entropy as the objective function. Therefore, the loss function is defined as follows:

$$\mathcal{L}_{\mathrm{MLE}} = -\sum_{s=1}^{S}\sum_{n=1}^{N^{(s)}}\log p(\mathbf{y}_n^{(s)}|\mathbf{y}_{<n}^{(s)},\mathbf{x}) \tag{4}$$

where $N^{(s)}$ is the length of $s-th$ target sentence and $p(\mathbf{y}_n^{(s)}|\mathbf{y}_{<n}^{(s)},\mathbf{x})$ is the probability of generating the $n-th$ word of $\mathbf{y}^{(s)}$ given by preceding ground-truth words and $\mathbf{x}^{(s)}$.

However, as pointed out by (Ranzato et al. 2016), model optimization by maximizing the likelihood of training data has two main drawbacks. One is the so-called "exposure bias": while models are trained only on the distribution of the training data, the

generation of target words at test time is conditioned to the previous model predictions. In this way, possible decoding errors at previous time steps will have significant impact on the following steps making the output generation diverge to completely wrong sentences (especially in the case of long input sentences). The other is that maximizing the probability of the next correct word based on the cross-entropy loss computed on previous word-level errors may have a low correlation with the metrics actually used for evaluation. To avoid these problems, better approaches should maximize the task-specific evaluation metrics that actually quantify translation quality. These, in the case of APE, are BLEU (Papineni et al. 2002) and TER (Snover et al. 2006). However, since these metrics are not parameterized w.r.t. to the model parameters, it is impossible to use common gradient descent for model optimization. This problem, which is well-known in MT, becomes particularly severe in APE, where automatic evaluation metrics against minimal post-edits (i.e. conservative "human-like" post edits) would penalise system's tendency to "over-correct" acceptable portions of the input sentence. Indeed, in APE, the goal is not to fully re-translate the MT output, which might certainly result in a good translation. Rather, the goal is to mimic post-editors' behaviour so to perform only the necessary corrections, leaving the correct parts of the MT output untouched. Maximizing BLUE (or minimizing TER) with respect to human post-edits will push the system towards a conservative correction strategy rather than the potentially more aggressive maximization of translations' probability.

In order to directly maximize these evaluation metrics, different approaches have been introduced, which mainly use reinforcement learning techniques. In RL methods for MT (Ranzato et al. 2016; Shen et al. 2016), the parameters of the MT system define a *policy* that chooses an *action*, i.e. generating a translation candidate $\hat{\mathbf{y}}$, and gets a *reward* $\Delta(\hat{\mathbf{y}})$ according to that action, i.e. a measure of output goodness based on the evaluation metric. The loss function in RL for MT is defined as expected reward, which has to be maximized:

$$\begin{aligned}\mathcal{L}_{\text{RL}} &= \sum_{s=1}^{S} E_{\hat{\mathbf{y}}\sim p(.|\mathbf{x}^{(s)})}\Delta(\hat{\mathbf{y}}) \\ &= \sum_{s=1}^{S}\sum_{\hat{\mathbf{y}}\in\mathbf{Y}} p(\hat{\mathbf{y}}|\mathbf{x}^{(s)})\Delta(\hat{\mathbf{y}})\end{aligned} \tag{5}$$

where $\mathbf{Y}$ is the set of all possible translation candidates. However, since the size of this set is exponentially large, it is practically impossible to compute the expected reward. To bypass this problem, in REINFORCE (Ranzato et al. 2016) the expected reward is estimated by random sampling only one candidate from this set.

$$\hat{\mathcal{L}}_{RL} = \sum_{s=1}^{S} p(\hat{\mathbf{y}}|\mathbf{x}^{(s)})\Delta(\hat{\mathbf{y}}), \hat{\mathbf{y}} \sim p(.|\mathbf{x}^{(s)}) \tag{6}$$

Alternatively, in Minimum Risk Training (MRT – (Shen et al. 2016)), this expected reward is estimated by subsampling a candidates set.

$$\hat{\mathcal{L}}_{RL} = \sum_{s=1}^{S}\sum_{\hat{\mathbf{y}}\in\mathcal{S}} q(\hat{\mathbf{y}}|\mathbf{x}^{(s)})\Delta(\hat{\mathbf{y}}) \tag{7}$$

Table 1
Statistics for the synthetic and in-domain datasets.

train			development	test	
synthetic 4M	synthetic 500K	in-domain	in-domain	in-domain 2016	in-domain 2017
4,391,180	526,368	23,000	1,000	2,000	2,000

where $q(\hat{\mathbf{y}}|\mathbf{x}^{(s)})$ is the normalized probability of the each hypothesis in the subsampled set $\mathcal{S}$:

$$q(\mathbf{y}|\mathbf{x}^{(s)}) = \frac{p(\mathbf{y}|\mathbf{x}^{(s)})}{\sum_{\mathbf{y}' \in \mathcal{S}} p(\mathbf{y}'|\mathbf{x}^{(s)})} \tag{8}$$

In the remainder of this paper, we will explore the application of both the methods to optimize our APE models on small in-domain data. Although these loss functions allow to directly optimize the model parameters to maximize task-specific evaluation metrics, recent studies on reinforcement learning for NMT have shown that their combination with MLE can yield further improvements (Wu et al. 2018). Moving a step forward, we will hence build on these findings and, in addition to RL-based loss functions, we will also experiment with their combination with MLE, so to obtain a combined loss function that considers both intrinsic word probabilities and extrinsic task-specific evaluation criteria:

$$\mathcal{L} = \mathcal{L}_{RL} + \mathcal{L}_{MLE} \tag{9}$$

4. Experiment Setup

In this section, we outline the data used for training and evaluating our APE system. Then, we describe the metrics used for evaluation. Finally, we give details about our baselines, the evaluated models and their specific setting.

4.1 Data

For the sake of a fair comparison with the best performing system at the WMT 2017 APE shared task (Chatterjee et al. 2017), we use the same training, development and test WMT datasets. The training data consists of three different corpora. One of them is released by the task organizers and contains 23K triplets from the Information Technology domain. The other two are synthetic data created by (Junczys-Dowmunt and Junczys-Dowmunt 2017). They respectively contain ∼4M and ∼500K English-German triplets generated by a round-trip translation process. In this process, a German-to-English and an English-to-German phrase-based translation models are respectively used to first translate German data into English, and then to translate the obtained output back into German. The original German monolingual data are considered as post-edits, the English translated data are considered as source sentences, and the German back-

translated data are considered as machine translation outputs. The development set is the one released for WMT 2017 APE shared task, which contains 1K in-domain triplets. We evaluate our models using the two test sets released for WMT 2016 and 2017 APE shared tasks, each containing 2K in-domain triplets. Table 1 summarizes the statistics of the datasets. To avoid unknown words and to keep under control the vocabulary size, we apply byte pair encoding (Sennrich, Haddow, and Birch 2016) to all the data using 32K rules.

4.2 Evaluation Metrics

For evaluation, we use the two official metrics of the WMT APE task, namely: *i)* TER (Snover et al. 2006) which is based on edit distance and *ii)* BLEU, which is the geometric mean of n-gram precision (Papineni et al. 2002). They are both applied on tokenized and true-cased data.

4.3 Terms of Comparison

We compare the performance of our Transformer models with two baselines.

Our first baseline (**MT Baseline**) is the official baseline used in the APE shared tasks; it mimics the behaviour of a "*do-nothing*" APE model that leaves all the original MT outputs untouched. In other words, leaving them unmodified, it reflects the quality of the original translations sent to APE.

The other baseline (**Ens8+RR**) is represented by the winning system at the WMT 2017 APE shared task (Chatterjee et al. 2017). This strong multi-component system comprises 4 different models based on the RNN architecture:

- `SRC_PE` a single-source model that exploits only the source sentence to generate the post-edits;
- `MT_PE` a single-source model that only exploits the machine translation output to generate the post-edits;
- `MT+SRC_PE` a multi-source model that exploits both the source sentence and the MT output to generate the post-edits;
- `MT+SRC_PE_TSL` another multi-source model with a task-specific loss function in order to avoid over correction.

For mixing the context vectors of the two encoders, Ens8 + RR uses a merging layer. This layer applies a linear transformation over the concatenation of the two context vectors. Chatterjee et al. (2017) compared the performance of these 4 models on the development set, and reported that MT+SRC_PE outperforms the other models. They also ensembled the two best models for each configuration to leverage all the models in a single decoder. On top of that, they also trained a re-ranker (Pal et al. 2017) to re-order the n-best hypotheses generated by this ensemble. In order to train the re-ranker, they used a set of features which are mainly based on edit distance. This set includes the number of insertions, deletions, substitutions and shifts, as well as the length ratios between MT output and APE hypotheses. It also includes precision and recall of the APE hypotheses. In Section 5, we compare our multi-source Transformer model with the ensembled model plus re-ranker (Ens8+RR). We train these models with the same

Table 2
Comparison between the Transformer-based APE and the best performing system at the WMT 2017 APE shared task.

Systems	Dev2017		Test2016		Test2017	
	TER (↓)	**BLEU** (↑)	**TER** (↓)	**BLEU**(↑)	**TER** (↓)	**BLEU** (↑)
MT Baseline	24.81	62.92	24.76	62.11	24.48	62.49
Ens8+RR	19.22	71.89	19.32	70.88	19.60	70.07
Transformer	**18.97**	**72.14**	**18.86**	**71.27**	**19.21**	**70.43**

settings reported in (Chatterjee et al. 2017) and, for more architecture details, we point to that paper.

4.4 System Setting

Similar to (Chatterjee et al. 2017), we initially train a generic **Transformer** model by using the ∼4M synthetic data. Then, for fine-tuning the resulting model, we try different combinations of the available data. First, in order to be comparable with (Chatterjee et al. 2017), we fine-tune the generic model on the union of the ∼500K and the in-domain training data (multiplied 20 times to give them more importance in the tuning process). Then, to analyze the contribution of each dataset, we also fine-tune the generic model separately on to ∼500K instances and on the in-domain data. As we will see in Section 5, the best performance is obtained by using only the in-domain data, which puts into perspective the "the more the better" assumption commonly shared by the APE task participants. In addition to MLE loss function, when fine-tuning the generic model only on in-domain data, we use the RL-based loss functions described in Section 3.2 (**REINFORCE** and **MRT**), both alone and in combination with MLE (**MLE+REINFORCE** and **MLE+MRT**).

Our Transformer model uses word embedding with 512 dimensions. The decoder and each encoder have 4 attention layers with 512 units, 4 parallel attention heads, and a feed-forward layer with 1,024 dimensions. For training the generic model, the model parameters are updated using the Lazy Adam optimizer (Kingma and Ba 2015), with mini-batch size of 8,192. The learning rate is varied using a warm-up strategy (Vaswani et al. 2017) with warm-up steps equal to 8,000. We use a Stochastic Gradient Descent optimizer with fixed learning rate to 0.5, and mini-batch size of 2,048 tokens. The dropout rate and the label smoothing value are set to 0.1. During decoding, we employ beam search with beam width equal to 10. For both the generic and fine-tuning steps, we continue the training for 40K steps and choose the best model checkpoints based on their performance on the development set. We use sentence-level BLEU in order to compute reward for each generated hypothesis, and for MRT we sample 5 different hypotheses. For our implementation, we use the OpenNMT-tf toolkit (Klein et al. 2017).

5. Results and Discussion

In order to have a fair comparison between our Transformer-based APE system and the best performing system in WMT 2017 APE shared task (Chatterjee et al. 2017), we fine-tuned our generic model trained on ∼4M synthetic data using the union of the ∼500K

Table 3
Performance of the multi-source Transformer-based APE, fine-tuned on different types of data, on the development set.

Systems	**TER** (↓)	**BLEU** (↑)
MT Baseline	24.81	62.92
Generic	29.64	57.46
Union	18.97	72.14
500K	26.28	61.26
in-domain	**18.49**	**72.23**

Table 4
Performance of the Transformer-based APE fine-tuned on in-domain data using different optimization approaches on development set

Systems	**TER** (↓)	**BLEU** (↑)
MT Baseline	24.81	62.92
MLE	**18.49**	72.23
REINFORCE	22.96	66.68
MRT	22.92	66.72
MLE+REINFORCE	18.61	**72.50**
MLE+MRT	18.53	**72.49**

and in-domain data (multiplied 20). Table 2 reports the results obtained by our system (Transformer) in comparison to the system by Chatterjee et al. (2017) (Ens8+RR). From the results, it is clear that our system (Transformer) outperforms (Ens8+RR) on all the datasets. These results confirm the superiority of our system, which relies on a single multi-source Transformer model that is far less complex than the state-of-the-art multi-component architecture previously deployed (an ensemble of 8 different RNN-based models, supported by an external re-ranking component).

To analyze the contribution of each dataset (∼500K synthetic and in-domain) to the improvement gained by fine-tuning, we separately fine-tuned the generic system on each dataset. As it is reported in Table 3, although fine-tuning only on ∼500K gains almost +15 BLEU points over the generic system, it is still lower than the MT Baseline. On the other hand, fine-tuning on in-domain data does not only outperform the MT Baseline, but it also achieves better performance than fine-tuning on the union of the two datasets. This analysis confirms that the assumption of "the more the better" in using data is not always true and, in contrast to the setting suggested by Chatterjee et al. (2017), the best performance can be obtained by fine-tuning on only in-domain data.

Furthermore, to maximize the exploitation of the in-domain data, we conducted a set of experiment using the two RL-based loss functions discussed in Section 3.2 (REINFORCE and MRT) alone and in addition to the MLE loss function. Table 4 reports the results obtained by using these loss functions on the development set. The results with both the metrics show that, although fine-tuning using only the RL-based loss

Table 5
Comparison between the multi-source Transformer-based APE and the best performing system of the WMT 2017 APE shared task on test sets of 2016 and 2017

Systems	Test2016		Test2017	
	TER (↓)	**BLEU** (↑)	**TER** (↓)	**BLEU** (↑)
MT Baseline	24.76	62.11	24.48	62.49
Ens8+RR	19.32	70.88	19.60	70.07
MLE	**18.73**	**71.54**	18.97	70.82
MLE+REINFORCE	18.74	71.46	**18.67**	71.12
MLE+MRT	18.82	71.38	18.75	**71.18**

functions leads to improvements over MT Baseline, their performance is still lower than the system fine-tuned using only MLE. To understand these results, we look at how the performance change during the fine-tuning process. The learning curve using RL-based loss functions is much lower than using MLE. Indeed, based on our observations during training, there is big jump in performance after few steps of fine-tuning on in-domain data using MLE, which is missing using RL-based loss functions. This is due to the fact that, in the RL loss functions, the reward is computed only once after generating the whole sequence, while MLE computes the loss after generating each word, resulting in a more coarse-grained feedback. Since, especially in industry settings, it is not cost-efficient to continue the training to reach to the maximum point using only RL-based losses, we prefer to use the RL-based loss functions together with MLE. Combining the two losses yields some improvements in terms of BLEU. Although these gains are probably no statistically significant, these results suggest that the integration of the evaluation metric in the optimization function can help the APE system.

In order to confirm our observations on the development set, we also evaluated our model fine-tuned on in-domain data using the different loss functions. Table 5 shows the results obtained on the two test sets. On the 2016 test set, the MLE loss function performs marginally better than MLE+REINFORCE and MLE+MRT. In contrast, on the 2017 test set, the addition of the RL-based loss functions to MLE (REINFORCE and MRT) outperforms, in both the cases, the system fine-tuned only using MLE.

To conclude, our experiments showed the superiority of our simple system based on a single multi-source Transformer model compared to the more complex RNN-based system by Chatterjee et al. (2017). We also demonstrated, in contrast to the previous approaches, that fine-tuning the system only on in-domain leads to better performance compared to augmenting the in-domain data with large amounts of synthetic data. Furthermore, by adding the RL-based loss functions (REINFORCE and MRT), the system can better exploit the gold in-domain data and reach better performance compared to the use of the MLE loss function alone.

6. Conclusion

In this paper, we approached the Automatic Post-Editing task focusing on effective solutions suitable for its industrial deployment. To this aim, in contrast to previous approaches, we developed a multi-source APE system that is: *i) fast to train* since it is

based on the Transformer architecture (the state of the art in MT, suitable for parallel processing) instead of RNNs, and *ii) easy to maintain* since it is based on one single model instead of multiple components. Extending our previous work (Tebbifakhr et al. 2018), we conducted a set of experiments to validate the assumption that, from the data standpoint, more data are always better for training a neural APE system. We found that this assumption is not always true. In particular, better performance can be obtained by fine-tuning the system only on gold (i.e. smaller, but higher quality) in-domain corpora. We also explored different training strategies to maximize the exploitation of small in-domain data. In particular, we used two different reinforcement learning techniques, namely REINFORCE and Minimum Risk Training, that allow us to optimize our model by considering task-specific evaluation metrics. Our experiments on the benchmark released for the WMT 2017 APE shared task show that, in a similar experimental setup, our system outperforms the best submissions to the shared task. Moreover, fine-tuning the system only on in-domain data and leveraging reinforcement learning techniques in combination with maximum likelihood leads to further improvements.

References

Aziz, Wilker, Sheila Castilho, and Lucia Specia. 2012. PET: a tool for post-editing and assessing machine translation. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3982–3987, Istanbul, Turkey, May. European Language Resources Association (ELRA).

Bahdanau, Dzmitry, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, July.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*, San Diego, California, USA, May.

Béchara, Hanna, 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, September.

Béchara, Hanna, 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 the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 215–230, Mumbai, India, December. The COLING 2012 Organizing Committee.

Bojar, Ondřej, 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*, pages 169–214, Copenhagen, Denmark, September. Association for Computational Linguistics.

Chatterjee, Rajen, 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*, pages 630–638, Copenhagen, Denmark, September. Association for Computational Linguistics.

Chatterjee, Rajen, Matteo Negri, Marco Turchi, Frédéric Blain, and Lucia Specia. 2018. Combining quality estimation and automatic post-editing to enhance machine translation output. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 26–38, Boston, MA, March. Association for Machine Translation in the Americas.

Chatterjee, Rajen, Marion Weller, Matteo Negri, and Marco Turchi. 2015. 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 (Volume 2: Short Papers)*, pages 156–161, Beijing, China, July. Association for Computational Linguistics.

Dugast, Loic, Jean Senellart, and Philipp Koehn. 2009. Statistical post editing and dictionary extraction: Systran/Edinburgh submissions for ACL-WMT2009. In *Proceedings of the Fourth*

Workshop on Statistical Machine Translation, pages 110–114, Athens, Greece, March. Association for Computational Linguistics.
Gehring, Jonas, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada, July. Association for Computational Linguistics.
Isabelle, Pierre, Cyril Goutte, and Michel Simard. 2007. Domain adaptation of mt systems through automatic post-editing. In *Proceedings of the 11th Machine Translation Summit*, pages 255–261, Copenhagen, Denmark, September.
Junczys-Dowmunt, Marcin 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, August. Association for Computational Linguistics.
Junczys-Dowmunt, Marcin and Marcin Junczys-Dowmunt. 2017. The AMU-UEdin submission to the WMT 2017 shared task on automatic post-editing. In *Proceedings of the Second Conference on Machine Translation*, pages 639–646, Copenhagen, Denmark, September. Association for Computational Linguistics.
Kalchbrenner, Nal, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, October.
Kingma, Diederik P. and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, California, USA, May.
Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July. Association for Computational Linguistics.
Knight, Kevin and Ishwar Chander. 1994. Automated postediting of documents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, pages 779–784, Seattle, Washington, USA, March. American Association for Artificial Intelligence.
Lagarda, Antonio-L., Vicent Alabau, Francisco Casacuberta, Roberto Silva, and Enrique Díaz-de Liaño. 2009. Statistical post-editing of a rule-based machine translation system. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 217–220, Boulder, Colorado, June. Association for Computational Linguistics.
Pal, Santanu, Sudip Kumar Naskar, Mihaela Vela, Qun Liu, and Josef van Genabith. 2017. Neural automatic post-editing using prior alignment and reranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 349–355, Valencia, Spain, April. Association for Computational Linguistics.
Pal, Santanu, 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: Short Papers)*, pages 281–286, Berlin, Germany, August. Association for Computational Linguistics.
Papineni, Kishore, 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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.
Ranzato, Marc'Aurelio, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations*, San Juan, Puerto Rico, May.
Rosa, Rudolf, David Mareček, and Ondřej Dušek. 2012. DEPFIX: A system for automatic correction of Czech MT outputs. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 362–368, Montréal, Canada, June. Association for Computational Linguistics.
Rosa, Rudolf, David Mareček, and Aleš Tamchyna. 2013. Deepfix: Statistical post-editing of statistical machine translation using deep syntactic analysis. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 172–179, Sofia, Bulgaria, August. Association for Computational Linguistics.
Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August.

Association for Computational Linguistics.
Shen, Shiqi, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany, August. Association for Computational Linguistics.
Simard, Michel, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. 2007. Rule-based translation with statistical phrase-based post-editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206, Prague, Czech Republic, June. Association for Computational Linguistics.
Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation of the Americas*, pages 223–231, Cambridge, Massachusetts, USA, August.
Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., Montreal, Quebec, Canada, December, pages 3104–3112.
Tebbifakhr, Amirhossein, Ruchit Agrawal, Matteo Negri, and Marco Turchi. 2018. Multi-source transformer for automatic post-editing. In *Proceedings of the Fifth Italian Conference on Computational Linguistics*, Torino, Italy, December.
Vaswani, Ashish, 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 30*. Curran Associates, Inc., Long Beach, California, USA, December, pages 5998–6008.
Wu, Lijun, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621, Brussels, Belgium, October-November. Association for Computational Linguistics.
Zoph, Barret and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California, June. Association for Computational Linguistics.