IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 19, 2022, accepted February 22, 2022, date of publication February 25, 2022, date of current version March 17, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3154768

RESHAPE: Reverse-Edited Synthetic Hypotheses for Automatic Post-Editing

WONKEE LEE[1], BAIKJIN JUNG[1], JAEHUN SHIN[1], AND JONG-HYEOK LEE[1,2]
[1]Department of Computer Science and Engineering, POSTECH, Pohang 37673, Republic of Korea
[2]Graduate School of Artificial Intelligence, POSTECH, Pohang 37673, Republic of Korea

Corresponding author: Wonkee Lee (wklee@postech.ac.kr)

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) funded by the Korean Government (MSIT) through the Artificial Intelligence Graduate School Program (POSTECH) under Grant 2019-0-01906.

ABSTRACT Synthetic training data has been extensively used to train Automatic Post-Editing (APE) models in many recent studies because the quantity of human-created data has been considered insufficient. However, the most widely used synthetic APE dataset, eSCAPE, overlooks respecting the minimal editing property of genuine data, and this defect may have been a limiting factor for the performance of APE models. This article suggests adapting back-translation to APE to constrain edit distance, while using stochastic sampling in decoding to maintain the diversity of outputs, to create a new synthetic APE dataset, **RESHAPE**. Our experiments show that (1) RESHAPE contains more samples resembling genuine APE data than eSCAPE does, and (2) using RESHAPE as new training data improves APE models' performance substantially over using eSCAPE.

INDEX TERMS Automatic post-editing, back-translation, decoding strategy, machine translation, synthetic data generation.

I. INTRODUCTION

Machine Translation (MT) has been developed to produce high-quality translations and is now being used in various areas. Nevertheless, MT is often inferior to a human translation, i.e., MT outputs may contain translation errors such as errors in lexical choice and word order, and these errors require post-editing to improve the original MT outputs. In this regard, Automatic Post-Editing (APE) has been proposed to improve the quality of given MT outputs by correcting errors or by tailoring the output style to a specific domain [1], [2]. Besides diminishing humans' post-editing efforts, APE is specifically beneficial when the MT system is given as a black-box because APE can revise the MT output on the fly without accessing the MT system's internal structure.

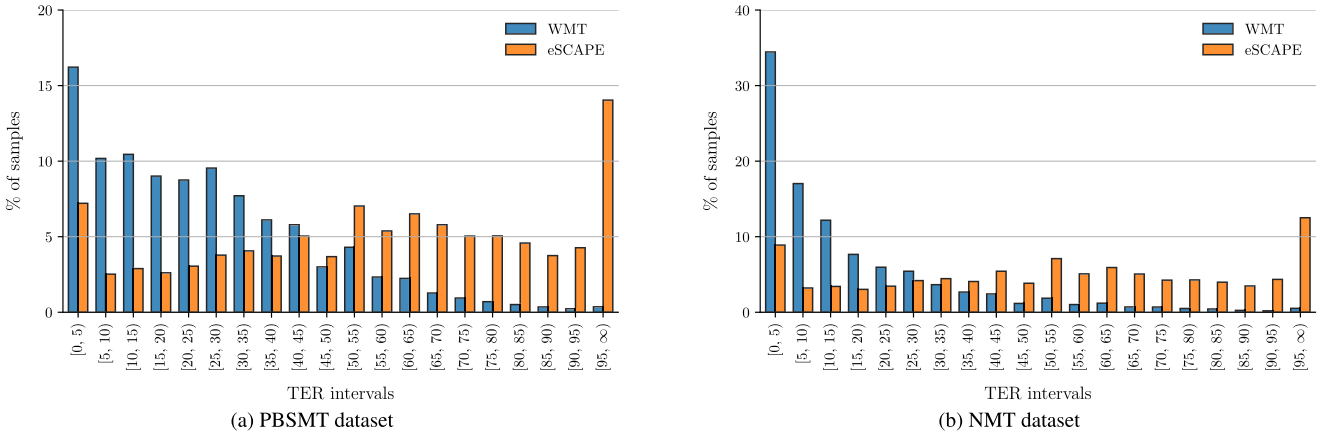
Most recent APE studies adopt neural multi-source sequence-to-sequence model architectures with supervised learning [2]–[4]. These models typically take a source text (*src*) and its MT output (*mt*) simultaneously as their inputs and take the post-edited text (*pe*) as their target. Thus, training those models requires triplet data, also called an **APE triplet**, that has the form of $\langle src, mt, pe \rangle$ (Fig. 1). Furthermore, APE data should satisfy the **minimum-editing criterion**, where *pe* should be created by **minimally editing** *mt* yet maintaining the meaning of *src*. However, the quantity of currently-available APE data is insufficient to train deep and complex APE models due to the high production cost of human post-edited data; this scarcity has become a limiting factor for the performance of APE models.

src	Manipulates the shape of an item .
mt	**Bearbeitet** die Form eines Elements **an** .
pe	**Verändert** die Form eines Elements .

FIGURE 1. An example of APE triplets from the English-to-German WMT APE dataset [2]. Boldface words are either incorrect words in *mt* or post-edited words in *pe*.

To mitigate the data scarcity, it has emerged as a possible solution to leverage synthetic data along with genuine data to expand the quantity of training data [6]–[8]. Especially, eSCAPE [7], a synthetic APE dataset made of parallel corpora, has been used extensively in many studies [2]–[4], [9], [10]. eSCAPE uses parallel corpora composed of *bitexts* — pairs of a source (*src*) and a reference (*ref*), to make a set of **synthetic APE triplets**: $\langle src, mt, ref \rangle$, in which *mt* is the MT output of *src*, and *ref* serves as *pe*.

The associate editor coordinating the review of this manuscript and approving it for publication was Davide Patti.



(a) PBSMT dataset (b) NMT dataset

FIGURE 2. (a) and (b) are categorical distributions that indicate the proportion of samples within a specific interval of Translation Error Rate (TER) [5] — a metric measuring edit distance between *mt* and its target (*pe* or *ref*). (a) describes the PBSMT datasets, *mt* of which are the outputs of phrase-based statistical MT (PBSMT) systems. (b) describes the NMT datasets, *mt* of which are the outputs of neural MT (NMT) systems. The WMT data is genuine data, and eSCAPE is synthetic data.

However, synthetic data usually lack certain qualities that genuine data retain. In the same vein, although eSCAPE can be an effective way to obtain large amounts of training data, it neglects to comply with the **minimum-editing criterion** that genuine APE data should follow. We note that *ref* is created independently of *mt* and is therefore not guaranteed to have been *minimally edited* from *mt*. Consequently, the correction patterns observed in this synthetic data may differ from those occurring in genuine data, and this violation results in a significant discrepancy in the distribution of edit distance between eSCAPE and genuine data (Fig. 2), possibly limiting the APE performance.

To solve this problem, we propose a new synthetic APE data-generation scheme that uses parallel corpora. We introduce **back-APE**[1] $(src, pe) \rightarrow mt$ which can be seen as an adaptation of back-translation [12] for APE. Back-APE learns to predict *mt* that exhibits the most likely error patterns for given *src* and *pe*. We expect that back-APE will produce erroneous hypotheses $\widetilde{mt}$ from parallel corpora based on the learnt error patterns. Eventually, we use $\widetilde{mt}$ and bitexts to compile a set of new synthetic triplets $\langle src, \widetilde{mt}, ref \rangle$ named "**R**everse-**E**dited **S**ynthetic **H**ypotheses for **A**utomatic **P**ost-**E**diting" (**RESHAPE**).

We further examine several decoding strategies for back-APE to identify which strategy yields synthetic data that lead to the biggest enhancement of the APE performance. Basically, decoding methods that maximize the model's output probabilities, such as beam search and greedy search, are the primary methods for sequence generation; although accurate, their outputs tend to be rather short and/or conservative [13]–[16]. We speculate that those decoding methods can confine back-APE output $\widetilde{mt}$ to certain error patterns, and thereby the generated training data can impede the model's learning. Thus, we suggest lending *randomness* with regard to the model probability to the decoding process of back-APE by adopting stochastic sampling methods to encourage the resulting synthetic samples to be diverse.

In our experiments, we use the same parallel corpora used to construct eSCAPE when constructing RESHAPE to make a fair comparison of our method with eSCAPE. For evaluation, we use the English-to-German WMT APE data [2], the *de-facto* standard benchmark. Experimental results demonstrate that, compared to eSCAPE, not only does our method improve the APE performance, but it also produces more samples with similar characteristics to genuine data.

II. PRELIMINARY: AUTOMATIC POST-EDITING

Fundamentally, APE has been recognized as a sequence-to-sequence learning problem and implemented by the sequence-to-sequence structure in which the encoder produces latent representations of a given source sequence and the decoder autoregressively produces the target sequence by taking the encoded representations. Due to the nature of APE that produces *pe* by revising *mt* while maintaining the meaning of *src*, the above-mentioned structure has been extended to **dual-source** (or multi-source) sequence-to-sequence structure $(src, mt) \rightarrow pe$ to accommodate two input sequences, in which *src* is treated as an auxiliary input providing contextual information, and *mt* serves as the primary input to be corrected.

The majority of recently proposed APE models adopt this dual-source structure that extends Transformer [17], and several variants of this structure have been proposed [18], [19]. Thanks to the article [19], which compared those variants to each other to identify the optimal architecture, we choose one dual-source architecture[2] (Fig. 3) that shows outstanding performance with fewer model parameters than others, as the

[1]This article is an extension of our published conference article [11], which has proposed the approach to adapting back-translation for APE. In this version, we additionally present (1) a method to make the training process of back-APE effective yet efficient, (2) an effective decoding strategy for back-APE, and (3) a more in-depth study.

[2]We emphasize that the model architecture is not the focus of this article; we pick one architecture for our experiments but expect others to achieve a similar result.

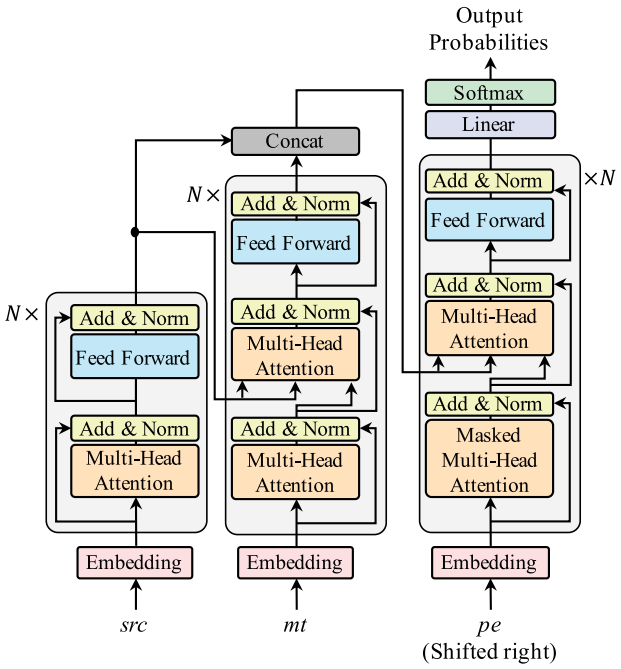


FIGURE 3. An illustration of the dual-source APE architecture [19] to be used in all our experiments.

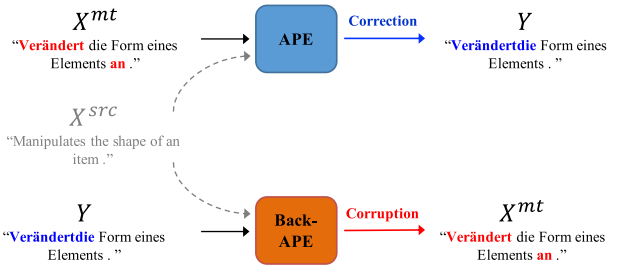


FIGURE 4. A schematic comparison between APE and back-APE.

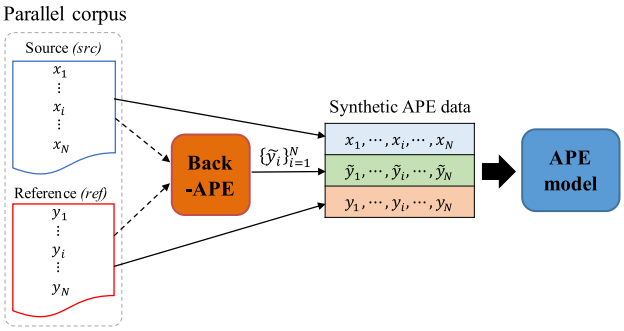


FIGURE 5. The overall construction process of RESHAPE.

underlying architecture shared by APE and back-APE in our experiments.

Schematically,[3] this APE model performs the following operations. Above all, we notate the set of natural numbers from 1 to n as $[n]$, i.e. $x_{[t]} = \{x_1, x_2, \ldots, x_t\}$ for notational convenience. First, let $\mathcal{D} = \{\langle \mathbf{x}^{src}, \mathbf{x}^{mt}, \mathbf{y}\rangle\}_n$ denote a set of training data, which is a collection of n APE triplets. For any training data $\langle \mathbf{x}^{src}, \mathbf{x}^{mt}, \mathbf{y}\rangle \in \mathcal{D}$, where $\mathbf{x}^{src} = x^{src}_{[T_{src}]}$, $\mathbf{x}^{mt} = x^{mt}_{[T_{mt}]}$, and $\mathbf{y} = y_{[T_{pe}]}$, in which T corresponds to the sequence length, *src* encoder first reads $\mathbf{x}^{src}$ to produce a sequence of representations

$$\mathbf{h}^{src} = \text{Encoder}_{src}(\mathbf{x}^{src}; \theta), \tag{1}$$

where $\mathbf{h}^{src} = h^{src}_{[T_{src}]}$. Then, *mt* encoder takes $\mathbf{x}^{mt}$ together with $\mathbf{h}^{src}$ to produce a sequence of contextualized representations

$$\mathbf{h}^{mt} = \text{Encoder}_{mt}(\mathbf{x}^{mt}, \mathbf{h}^{src}; \theta), \tag{2}$$

where $\mathbf{h}^{mt} = h^{mt}_{[T_{mt}]}$. For every *pe* $y_i \in \mathbf{y}$, the decoder autoregressively produces its conditional probability as

$$\begin{aligned} h^{pe}_i &= \text{Decoder}(y_{[i-1]}, \mathbf{h}^{src}, \mathbf{h}^{mt}; \theta), \\ P(y_i | y_{[i-1]}, \mathbf{x}^{src}, \mathbf{x}^{mt}) &\propto \text{Softmax}(h^{pe}_i \mathbf{W}^{pe}), \end{aligned} \tag{3}$$

where $\mathbf{W}^{pe} \in \mathbb{R}^{d \times |V|}$, in which d and $|V|$ denotes the sizes of the hidden dimension and the vocabulary, respectively. Finally, the model is trained to minimize the negative log-likelihood of the output probabilities with the following objective function

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} \sum_{i=1}^{T_{pe}} \log \left(P(y_i \mid y_{[i-1]}, \mathbf{x}^{src}, \mathbf{x}^{mt}) \right). \tag{4}$$

III. ADAPTATION OF BACK-TRANSLATION TO APE

Back-translation [12] is a method widely used in MT to obtain additional parallel resources by leveraging monolingual texts. Back-translation first trains an MT system in the reverse direction, from the target to the source, and then uses the trained MT system to generate synthetic source-side texts from the target-side monolingual texts. Motivated by that method, we introduce back-APE; we apply the back-translation method to an APE model and use it to create new synthetic APE triplets by leveraging parallel corpora.

The back-APE method reverses the original APE process by swapping the positions of *mt* and *pe*, thus aiming to learn (*src*, *pe*) → *mt*.[4] Consequently, back-APE trains to minimize the following objective function with a given dataset $\mathcal{D} = \{\langle \mathbf{x}^{src}, \mathbf{y}, \mathbf{x}^{mt}\rangle\}_n$ (cf. Equation (4)):

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} \sum_{j=1}^{T_{mt}} \log \left(P(x^{mt}_j \mid x^{mt}_{[j-1]}, \mathbf{x}^{src}, \mathbf{y}) \right). \tag{5}$$

[3] To avoid clutter, we omit detailed operations of Transformer such as the multi-head attention, position-wise feed-forward network, layer normalization and residual connection. We refer readers to the original papers [17] for more details.

[4] Note that such modeling makes a difference from our previous work in which back-APE was constructed in the form of (*src*, *ref*) → *mt*. Assuming that back-APE (*src*, *ref*) → *mt* is trained until convergence, the results are expected to be very similar to the original *mt* and thus may not be meaningful (since the results may not reduce the edit distance from *ref* drastically). Thus, our previous study relied on empirical observations to determine its training stop point, leading to inefficient learning.

Thus, whereas APE outputs a minimally 'corrected' text (*pe*) from *mt* while considering the meaning of *src*, back-APE outputs a minimally 'corrupted' text (*mt*) from *pe* conditioned on *src* (Fig. 4), i.e., back-APE can be interpreted as learning to produce *mt* that is likely to correspond with the given *src* and *pe*.

Accordingly, when supplying a bitext (*src*, *ref*) to back-APE at inference time, we expect back-APE to produce $\widetilde{mt}$ whose error patterns (distribution) are influenced by genuine APE data and also expect that $\widetilde{mt}$ and *ref* follow the minimum-correction principle. Subsequently, by using back-APE's output ($\widetilde{mt}$) and its input bitext, we construct a new set of synthetic APE triplets $\langle src, \widetilde{mt}, ref \rangle$ RESHAPE (Fig. 5). Note that letting *src* and *ref* serve as *src* and *pe* respectively can be reasonable because each side of parallel corpora is supposed to be error-free and semantically equivalent to each other; likewise, *src* and *pe* of a genuine APE triplet have the same relation.

IV. DECODING STRATEGIES FOR BACK-APE

A. MAXIMUM-A-POSTERIORI DECODING

Besides the back-APE training, we consider several decoding strategies to discover which produces synthetic samples that provide the most significant learning effect. First, we can typically consider maximum-*a-posteriori* (MAP) decoding strategies, which find the most likely sequences by maximizing the model probabilities, i.e., MAP yields

$$\begin{aligned}\mathbf{x}^{mt} &= \arg\max P(x^{mt}_{[T]} \mid \mathbf{x}^{src}, \mathbf{y}) \\ &= \arg\max \prod_{t=1}^{T} P(x^{mt}_{t} \mid x^{mt}_{[t-1]}, \mathbf{x}^{src}, \mathbf{y}). \end{aligned} \tag{6}$$

However, the complete MAP decoding is almost intractable due to its inevitably vast search space $O(|V|^m)$, where V is the vocabulary set, and m is the maximum sequence length. Therefore, MAP is usually approximated by *beam search* or *greedy search*. Beam search is a best-first search algorithm that traces only the B most likely prefixes $\mathcal{X}^{mt}_{t-1} = \{x^{mt}_{[t-1],b}\}_{b=1}^{B}$ (where B and t denote the beam size and the decoding time step, respectively). In detail, at every time step t, every possible token $x^{mt}_{t} \in V$ is added to each prefix in $\mathcal{X}^{mt}_{t-1}$, advancing to $\hat{\mathcal{X}}^{mt}_{t} = \left\{x^{mt}_{[t-1],b} \cup x^{mt}_{t} \mid \forall\, b \in [1, B], x^{mt}_{t} \in V\right\}$, and then, top-$B$ prefixes for the next time step are selected as

$$\mathcal{X}^{mt}_{t} = \underset{\substack{\mathcal{X} \subset \hat{\mathcal{X}}^{mt}_{t} \\ |\mathcal{X}|=B}}{\arg\max} \sum_{x^{mt}_{[t]} \in \mathcal{X}} P(x^{mt}_{[t]} \mid \mathbf{x}^{src}, \mathbf{y}) \tag{7}$$

reducing the search space to $O(|V|B^m)$. Meanwhile, greedy search simply picks the t-th token with the highest model probability:

$$x^{mt}_{t} = \arg\max P(x^{mt}_{t} \mid x^{mt}_{[t-1]}, \mathbf{x}^{src}, \mathbf{y}) \tag{8}$$

at every time step t. Under the assumption that the probability assigned by the model increases as the quality of the output increases, they are commonly used for many sequence-generation problems; thus, we can consider adopting them for back-APE as well.

Nevertheless, inadequacies of the MAP decoding have also been reported [13]–[15]. Because MAP determines its output by taking just the modes in the model distribution, MAP may not reproduce various other statistics of the training data, resulting in conservative, generic, and/or relatively uninformative outputs. In addition, favoring high sentence scores, MAP tends to underestimate the sentence length. Although such phenomena are known to be particularly problematic for tasks that aim to generate human-like texts, such as dialogue generation [13] and story generation [14]–[16], we speculate that such downsides could also be problematic for back-APE because favoring only the top-scoring texts may result in highly homogeneous outputs and thus restrict the diverse error patterns (statistics) found in genuine APE data. Consequently, the lack of diversity in synthetic data can lead to biased learning effects.

B. SAMPLING METHOD

Sampling methods, in which each output token is determined *stochastically* according to the model distribution, are known to better represent various statistics of the training data into the output; sampling methods preclude the model from taking only the modes of its distribution and estimate the targeted distribution more completely than MAP methods. When sampling is adopted in back-APE in the decoding stage, the model selects the t-th token *randomly* at every decoding time step with weight given by the model distribution, i.e.,

$$x^{mt}_{t} \sim P(x^{mt}_{t} \mid x^{mt}_{[t-1]}, \mathbf{x}^{src}, \mathbf{y}). \tag{9}$$

For back-APE, it appears promising to use sampling methods to yield synthetic training data that reflects various statistics of genuine APE data. Sampling methods are expected to generate $\widetilde{mt}$ by considering all possible error patterns that are likely to appear in the given input bitext based on the learnt statistics, so the generated samples will provide diverse learning patterns to APE models. In some cases, however, sampling methods may produce arbitrarily poor outputs that are inconsistent and/or context-independent if they are frequently drawn from the tails of the model distribution where tokens are assigned relatively low, but non-zero, probabilities. Therefore, certain measures that help bound the error of model outputs are required.

C. RESTRICTED SAMPLING METHOD

Restricted sampling methods [14], [15] have been proposed to rectify the unboundedness problem that arises from sampling methods by excluding the unreliable tails of the model distribution. Especially, top-k sampling [14], a straightforward but powerful scheme, has recently become a popular method to obtain high-quality texts in many human-like text generation tasks [14]–[16]. Top-k sampling samples within the k most likely tokens at each decoding step, presuming that the tokens not ranked as the top-k are unreliable; top-k sampling can be regarded as a compromise between MAP

and pure sampling. Thus, at every time step, the model distribution is trimmed to contain only the probabilities of the top-k tokens.

Provided that we apply top-k sampling to back-APE, the original model distribution will be re-scaled to $\tilde{P}(\cdot)$ at every decoding step t referring to the top-k vocabulary $V_t^k \subset V$. Specifically,

$$\tilde{P}(x_t^{mt} \mid x_{[t-1]}^{mt}, \mathbf{x}^{src}, \mathbf{y}) = \begin{cases} \dfrac{P(x_t^{mt} \mid x_{[t-1]}^{mt}, \mathbf{x}^{src}, \mathbf{y})}{\sum_{x_t^{mt} \in V_t^k} P(x_t^{mt} \mid x_{[t-1]}^{mt}, \mathbf{x}^{src}, \mathbf{y})} & \text{if } x_t^{mt} \in V_t^k \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where

$$V_t^k = \underset{\substack{V^k \subset V \\ |V^k|=k}}{\arg\max} \sum_{x_t^{mt} \in V^k} P(x_t^{mt} \mid x_{[t-1]}^{mt}, \mathbf{x}^{src}, \mathbf{y}). \tag{11}$$

The t-th token will be subsequently sampled from the following re-normalized distribution

$$x_t^{mt} \sim \tilde{P}(x_t^{mt} \mid x_{[t-1]}^{mt}, \mathbf{x}^{src}, \mathbf{y}). \tag{12}$$

We expect that applying top-k sampling to back-APE will reduce the noise from pure sampling yet still allow the outputs to reflect the properties of the error distribution of the training data.

V. EXPERIMENT

A. SETTINGS

1) EVALUATION METRIC

As in the WMT APE shared task [2], [3], we used the *de-facto* standard evaluation metrics: TER[5] [5], a primary metric that measures the edit distance from the model hypotheses to its target; BLEU[6] [20], a secondary metric that measures the degree of n-gram match between the model hypotheses and its target. All of our evaluations are case-sensitive.

2) DATASETS

We started with the WMT English–German (EN–DE) APE dataset, which is the *de-facto* standard APE benchmark, coming from the IT domain [2], [3]. The WMT data were used to train both back-APE and APE, and to evaluate the APE performance. We also used EN–DE eSCAPE (1) to train back-APE due to the small size of the WMT dataset; (2) to construct RESHAPE by feeding its (*src*, *ref*) into the trained back-APE model for a fair comparison; and (3) to train one of the baseline APE models. All of our APE data are also categorized by their subtasks: whether *mt* has been produced by either a phrase-based statistical MT (PBSMT) system or a neural MT (NMT) system; detailed data statistics are presented in Table 1. We tokenized words into subword units by using SentencePiece.[7]

[5]https://github.com/jhclark/tercom
[6]https://github.com/moses-smt/mosesdecoder
[7]https://github.com/google/sentencepiece

TABLE 1. Statistics of the WMT and eSCAPE datasets on the PBSMT and NMT subtasks. The WMT dataset contains three different testsets for the PBSMT task yet contains only one testset for the NMT task.

Dataset		# Triplets
WMT-PBSMT	Train	23,000
	Dev	1,000
	Test2016	2,000
	Test2017	2,000
	Test2018	2,000
WMT-NMT	Train	13,442
	Dev	1,000
	Test2018	1,023
eSCAPE-PBSMT		7,258,533
eSCAPE-NMT		7,258,533

3) MODEL CONFIGURATION

We used OpenNMT-py[8] to implement the aforementioned dual-source architecture (§II). We adopted the "base Transformer settings" [17] for all the models: specifically, 512 for all the hidden dimensions including the embedding dimensions, 2048 for the feed-forward layers, 6 layers, 8 attention heads, a dropout probability of 10%, a label smoothing value of 0.1, the Adam [21] optimizer with $\beta = (0.9, 0.998)$, and 6,000 warm-up steps followed by an inverse square root decay of learning rate. For the back-APE and APE training, we pre-trained the models with a batch size of 48K tokens on the combination of the synthetic data (either eSCAPE or RESHAPE) and the WMT data and subsequently fine-tuned them with a batch size of 1024 tokens by using only the WMT data. To evaluate the APE models, we used beam search with a beam size of 6 to obtain their hypotheses.

B. BACK-APE TRAINING AND RESHAPE CONSTRUCTION

We used eSCAPE both to pre-train back-APE and to construct RESHAPE, so we applied the 'n-fold cross-generation technique' (adapting n-fold cross-validation) to back-APE to avoid producing biased outputs for which the inputs have already been used during training. Specifically,

1) Split eSCAPE into $n = 4$ folds: $\{d_i\}_{i=1}^4$.
2) For each of n back-APE models $\{\mathcal{M}_i\}_{i=1}^n$, construct a training dataset $\mathcal{D}_i$ with $n-1$ folds: $\mathcal{D}_i = \bigcup_{j=1}^4 d_j \setminus d_i$.
3) Pre-train $\mathcal{M}_i$ using the combination of $\mathcal{D}_i$ and the WMT training set, and then fine-tune $\mathcal{M}_i$ only on the WMT training set until the validation loss on the WMT development set converges.
4) Obtain $\langle src, \widetilde{mt}, ref \rangle \in \text{RESHAPE}_i$ by supplying the model $\mathcal{M}_i$ with input bitexts (*src*, *ref*) in the corresponding held-out fold d_i, to obtain $\widetilde{mt}$.
5) Finally, construct $\text{RESHAPE} = \bigcup_i \text{RESHAPE}_i$.

When generating $\widetilde{mt}$, we used a beam size of 6 for beam search decoding; in the case of top-k sampling, we attempted

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

TABLE 2. Comparison of decoding methods in generating RESHAPE. Each row represents the evaluation results of the APE models trained on RESHAPEs, each of which is generated by a different decoding method. 'Test Avg.' in the PBSMT task refers to the average of the evaluation results for all testsets.

Decoding methods	PBSMT task				NMT task			
	Dev		Test Avg.		Dev		Test2018	
	TER(↓)	BLEU(↑)	TER(↓)	BLEU(↑)	TER(↓)	BLEU(↑)	TER(↓)	BLEU(↑)
Beam	18.01	73.38	18.04	72.17	15.24	76.94	16.73	75.08
Greedy	17.45	73.90	17.56	72.79	14.92	77.42	16.48	75.50
Sampling	17.18	74.55	17.31	73.29	14.87	**77.70**	15.98	**76.35**
Top-*k* sampling	**16.98**	**74.62**	**17.07**	**73.47**	**14.73**	77.69	**15.96**	76.32

TABLE 3. The evaluation results of the APE models trained on different training-data configurations. 'Test Avg.' in the PBSMT task refers to the average of the evaluation results for all testsets. '†' indicates that the model's improvement upon the eSCAPE baseline (the strongest baseline) is statistically significant ($p \leq 0.05$). The last two rows are current top models. The best result among our models in each column is in bold type.

Settings	PBSMT task										NMT task			
	Dev		Test2016		Test2017		Test2018		Test Avg.		Dev		Test2018	
	TER(↓)	BLEU(↑)	TER(↓)	BLEU(↑)	TER(↓)	BLEU(↑)	TER(↓)	BLEU(↑)	TER(↓)	BLEU(↑)	TER(↓)	BLEU(↑)	TER(↓)	BLEU(↑)
NO EDIT	24.81	62.92	24.76	62.11	24.48	62.49	24.24	62.99	24.49	62.53	15.08	76.76	16.84	74.73
WMT only	28.70	59.54	27.10	60.57	27.23	59.42	27.15	59.41	27.16	59.80	19.71	69.55	21.05	68.33
eSCAPE	17.43	73.95	16.79	74.24	17.31	73.01	17.54	72.65	17.21	73.30	14.94	77.62	16.26	76.06
RESHAPE	**16.98**†	**74.62**†	**16.70**	**74.26**	**17.07**†	**73.38**†	**17.43**	**72.77**	**17.07**†	**73.47**†	**14.73**†	**77.69**	**15.96**†	**76.32**†
CopyNet-APE [9]	18.38	72.99	17.45	73.51	17.77	72.98	–	–	–	–	14.88	77.40	–	–
BERT-APE [10]	–	–	16.91	74.29	17.26	73.42	17.71	72.74	17.29	73.48	–	–	–	–

various k values to investigate the training effect of $\widetilde{mt}$ at different k values.

VI. RESULTS

A. EXAMINATION OF BACK-APE DECODING SCHEMES

We prepared four RESHAPE datasets, each by using one of the back-APE decoding schemes (beam search, greedy search, sampling, and top-k sampling). To compare their learning effects to each other, we have trained APE models on each RESHAPE and then evaluated their performance (Table 2); in the case of top-k sampling, among various k values (Fig. 6), we only report the best one in Table 2.

Firstly, we observed that making use of sampling methods (either pure sampling or top-k sampling) results in consistent improvements in the APE performance over those of MAP decoding methods. Among the sampling methods, top-k sampling led to better APE performance than pure sampling in most cases. However, we found that pure sampling is competitive with top-k sampling in the NMT task, although top-k sampling consistently yields better TER than pure sampling. Our speculation on this phenomenon is that the edit-distance distributions of the two tasks have different skewnesses. In contrast with the PBSMT dataset, the edit-distance distribution of the NMT dataset is drastically skewed (Fig. 2): most of the samples have only a small number of errors, and therefore the output probabilities of back-APE are likely to be concentrated on just a few candidates, possibly making not much different distribution between pure sampling and top-k sampling.

B. QUANTITATIVE APE EVALUATION

According to the previous subsection, we adopted RESHAPE created by applying top-k sampling to the back-APE as our final result for new synthetic training APE data. To evaluate the APE performance, we considered three baselines:

- NO EDIT: a standard baseline formed by the evaluation of the raw *mt*, which has not yet been post-edited, in the test datasets, indicating the initial margin for improvement to be achieved by APE.
- WMT only: an APE model trained solely on the WMT data.
- eSCAPE: an APE model trained on the WMT data augmented with eSCAPE, which can be regarded as our main baseline.

The evaluation results (Table 3) demonstrate that our approach surpasses the first two baselines (i.e. NO EDIT

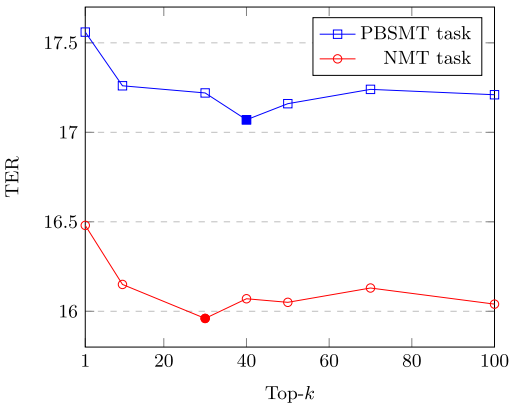


FIGURE 6. The effect of k for top-k sampling. y-axis represents the evaluation result of the APE model that is trained on RESHAPE using the top-k sampling decoding method where $k = x$. The evaluation was conducted on the WMT testsets (three TER results were averaged for the PBSMT task). The k that records the best performance is marked with a color ($k = 40$ for the PBSMT subtask and $k = 30$ for the NMT subtask).

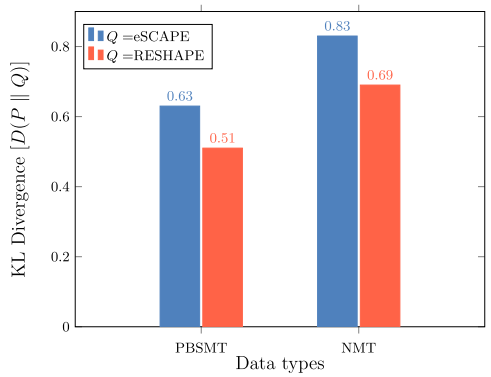


FIGURE 7. *D*(*P* ‖ *Q*) (with natural logarithms) where *P* and *Q* are edit-distance distributions; *P* is for the WMT data, and *Q* is for either RESHAPE or eSCAPE.

TABLE 4. The *k*-nearest neighbor results with varying *k*. We count the number of neighbors of each data point in the WMT data, which are *k* data points in either RESHAPE or eSCAPE.

Tasks	*k*	# of *k*-nearest triplets	
		RESHAPE	eSCAPE
PBSMT	1	17,409 (58.03%)	12,591 (41.97%)
	3	52,641 (58.49%)	37,359 (41.51%)
	5	88,176 (58.78%)	61,824 (41.22%)
	7	124,037 (59.07%)	85,963 (40.93%)
	9	159,527 (59.08%)	110,473 (40.92%)
NMT	1	10,630 (68.74%)	4,835 (31.26%)
	3	30,833 (66.46%)	15,562 (33.54%)
	5	50,863 (65.78%)	26,462 (34.22%)
	7	71,288 (65.85%)	36,967 (34.15%)
	9	91,290 (65.59%)	47,895 (34.41%)

and `WMT only`) by a substantial margin, indicating that augmenting gold APE data (the WMT data) with RESHAPE is remarkably beneficial. More importantly, we observed that our approach outperformed the `eSCAPE` baseline in all the test cases, even achieving a statistically significant gain in most cases. This result suggests that RESHAPE is more advantageous than eSCAPE to APE learning.

Our approach also outperformed the state-of-the-art APE models: `CopyNet-APE` [9] and `BERT-APE` [10]. Both are Transformer-based APE models and were trained on eSCAPE; `CopyNet-APE` adds CopyNet [22] layers, and `BERT-APE` adopts BERT [23] weights for the APE architecture. It is particularly noticeable that our APE model trained on RESHAPE even outperformed `BERT-APE`, which contains a larger number of model parameters and was pre-trained on tens of millions of monolingual data.

VII. DISCUSSION

Our quantitative results reveal that RESHAPE is better than eSCAPE. Beyond that, we further discuss and analyze our results from the various perspectives on whether RESHAPE is more similar to human-made APE data than eSCAPE is.

We have pointed out at the beginning (§I) that eSCAPE and the WMT data differ greatly in terms of the edit-distance distribution (Fig. 2), so we first examined whether RESHAPE helps reduce such discrepancy. For this purpose, we compiled the TER statistics for the WMT data,[9] eSCAPE, and RESHAPE in the same manner as in Fig 2, then measured the KL-divergence of the TER distributions of eSCAPE and RESHAPE against the WMT data. As a result, RESHAPE showed a smaller divergence than eSCAPE (Fig. 7); this result indicates that the edit-distance distribution of RESHAPE is more similar to that of the WMT data, compared to the eSCAPE's distribution.

Apart from the edit distance, we verified how similar RESHAPE is to the WMT data in various aspects. We empirically designed 14 features to characterize an APE triplet as follows:

- A TER score and the rates of four editing operations (insertion, deletion, substitution, and shift) in the TER calculation.
- Lengths of each triplet element.: T_{src}, T_{mt}, T_{pe}.
- Length ratios between two elements which have an 'input-output' relation: T_{mt}/T_{src}, T_{pe}/T_{src}, T_{pe}/T_{mt}.
- Language-model scores of each triplet element. Each element was scored by its corresponding language model (LM_{src}, LM_{mt}, LM_{pe}), each of which is a 5-gram Kneser-Ney language model [24] trained on the corresponding elements in the WMT data.

We then represented all triplets included in RESHAPE, eSCAPE, and the WMT data as 14-dimensional vectors with these features, excluding triplets that are represented as the same feature vector for both RESHAPE and eSCAPE as no comparison is possible. Finally, for each WMT triplet, we used the *k*-nearest neighbor algorithm (calculated by Euclidean distance, after normalization of each feature) to search for the *k* closest synthetic triplets, either in RESHAPE or eSCAPE; we then counted how many come from each. We found that the *k*-nearest neighborhoods that belong to RESHAPE outnumbers those of eSCAPE (Table 4); this result supports our speculation that RESHAPE is capable to capture various characteristics of genuine APE data better than eSCAPE.

Finally, we present some examples from RESHAPE and eSCAPE (Table 5). We found that *mt* in eSCAPE tend to undergo an excessive number of corrections, much more than the expected number of corrections for the WMT data, whereas *mt* in RESHAPE require fewer corrections, which is in turn similar to that of the WMT data.

[9]The entire WMT dataset. i.e., the union of the training, development, and test datasets.

TABLE 5. Examples of synthetic triplets. *src* and *pe* are shared by eSCAPE and RESHAPE. Boldface words in *mt* need to be corrected to match *pe*. TER in the *mt* rows are calculated with regard to *pe*.

src	On that occasion it was decided that eleven Member States would join the zone .
pe (= *ref*)	Damals wurde entschieden , dass elf Mitgliedstaaten der Eurozone beitreten .
mt (eSCAPE)	**Bei dieser Gelegenheit** wurde **beschlossen** , dass elf Mitgliedstaaten der **Euro-Zone** beitreten **würden** . (TER = 54.54)
mt (RESHAPE)	Damals wurde **beschlossen** , dass elf Mitgliedstaaten der Eurozone beitreten **würde** . (TER = 18.18)
src	I am not sure that the Greenlandic , Russian and Canadian Inuits would agree with that .
pe (= *ref*)	Ich bin nicht sicher , ob die grönländischen , russischen und kanadischen Inuit damit einverstanden wären .
mt (eSCAPE)	Ich bin **mir** nicht sicher , **dass** die **Inuit** grönländischen , **Russisch** und **in Kanada würden dem zustimmen** . (TER = 52.95)
mt (RESHAPE)	Ich bin **mir** nicht sicher , ob die grönländischen , russischen und kanadischen **Inuits** einverstanden wären . (TER = 17.65)

VIII. CONCLUSION

In this article, we introduce a new synthetic APE dataset, RESHAPE, derived from parallel corpora. We summarize our research findings as follows:

1) We propose the back-APE method, which adapts the back-translation technique for APE, to produce synthetic data that captures inherent characteristics of genuine APE data, and this method is a refined version from our previous study [11].
2) We investigate several decoding methods for back-APE and conclude that sampling methods, top-k sampling especially, are superior to MAP decoding methods.
3) Through our quantitative and qualitative evaluation, we observed that compared to the existing method, eSCAPE, our method produces more synthetic samples containing similar characteristics to genuine APE data, including the edit-distance properties, which in turn contribute to improving the APE performance.
4) Finally, we believe that our findings present the importance of reflecting the characteristics of genuine data in synthetic data.

Considering that this is the first attempt at adapting back-translation for APE, we believe that our findings suggest further research directions. First, because this work is simply employing one APE architecture without any modification for back-APE, several extensions such as modifying the training objective function or the architecture in itself will be meaningful future studies. Second, we found that although back-APE helps produce triplets similar to gold APE data, the edit distances of RESHAPE and the WMT data (Fig. 7) still differ. Therefore, a useful study would be to explore a way to control the quantity of the errors to be injected, by following the error distribution of genuine data.

ACKNOWLEDGMENT

The authors would like to thank their colleague Aren Siekmeier for taking the time to proofread this article.

REFERENCES

[1] K. Parton, N. Habash, K. McKeown, G. Iglesias, and A. de Gispert, "Can automatic post-editing make MT more meaningful," in *Proc. 16th Annu. Conf. Eur. Assoc. Mach. Transl.*, 2012, pp. 111–118.
[2] R. Chatterjee, M. Negri, R. Rubino, and M. Turchi, "Findings of the WMT 2018 shared task on automatic post-editing," in *Proc. 3rd Conf. Mach. Transl., Shared Task Papers*, 2018, pp. 723–738.
[3] R. Chatterjee, C. Federmann, M. Negri, and M. Turchi, "Findings of the WMT 2019 shared task on automatic post-editing," in *Proc. 4th Conf. Mach. Transl. (Shared Task Papers, Day)*, vol. 3, 2019, pp. 11–28.
[4] R. Chatterjee, M. Negri, R. Rubino, and M. Turchi, "Findings of the WMT 2018 shared task on automatic post-editing," in *Proc. 3rd Conf. Mach. Transl., Shared Task Papers*, 2018, pp. 646–659.
[5] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, "A study of translation edit rate with targeted human annotation," in *Proc. 7th Conf. Assoc. Mach. Transl. Americas*, 2006, pp. 223–231.
[6] M. Junczys-Dowmunt and R. Grundkiewicz, "Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing," in *Proc. 1st Conf. Mach. Transl., Shared Task Papers*, vol. 2, 2016, pp. 751–758.
[7] M. Negri, M. Turchi, R. Chatterjee, and N. Bertoldi, "ESCAPE: A large-scale synthetic corpus for automatic post-editing," in *Proc. 11th Int. Conf. Lang. Resour. Eval.*, 2018, pp. 24–30.
[8] W. Lee, J. Shin, B. Jung, J. Lee, and J.-H. Lee, "Noising scheme for data augmentation in automatic post-editing," in *Proc. 5th Conf. Mach. Transl.*, 2020, pp. 783–788.
[9] X. Huang, Y. Liu, H. Luan, J. Xu, and M. Sun, "Learning to copy for automatic post-editing," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 6122–6132.
[10] G. M. Correia and A. F. T. Martins, "A simple and effective approach to automatic post-editing with transfer learning," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 3050–3056.
[11] W. Lee, B. Jung, J. Shin, and J.-H. Lee, "Adaptation of back-translation to automatic post-editing for synthetic data generation," in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics, Main Volume*, 2021, pp. 3685–3691.
[12] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2016, pp. 86–96.
[13] I. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, 2016, pp. 3776–3783.
[14] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2018, pp. 889–898.
[15] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–3.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI Blog, 2019.
[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[18] F. do Carmo, D. Shterionov, J. Moorkens, J. Wagner, M. Hossari, E. Paquin, D. Schmidtke, D. Groves, and A. Way, "A review of the state-of-the-art in automatic post-editing," *Machine Transl.*, vol. 35, no. 2, pp. 101–143, 2020.
[19] J. Shin, W. Lee, B.-H. Go, B. Jung, Y. Kim, and J.-H. Lee, "Exploration of effective attention strategies for neural automatic post-editing with transformer," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 20, no. 6, pp. 1–17, Nov. 2021.
[20] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2001, pp. 311–318.
[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[22] J. Gu, Z. Lu, H. Li, and V. O. K. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2016, pp. 1631–1640.
[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.
[24] K. Heafield, I. Pouzyrevsky, J. H. Clark, and P. Koehn, "Scalable modified Kneser–Ney language model estimation," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics*, 2013, pp. 690–696.

WONKEE LEE received the B.S. degree in computer science from Dongguk University, Seoul, South Korea, in 2016. He is currently pursuing the integrated M.S. and Ph.D. degree with POSTECH, Pohang, South Korea. His research interests include natural language processing, machine learning, and artificial intelligence.

BAIKJIN JUNG received the B.S. degrees in German language and literature, and cognitive sciences from Yonsei University, Seoul, South Korea, in 2019. He is currently pursuing the integrated M.S. and Ph.D. degree with POSTECH, Pohang, South Korea. His research interests include natural language processing, machine learning, and artificial intelligence.

JAEHUN SHIN received the B.S. and Ph.D. degrees in computer science from POSTECH, Pohang, South Korea, in 2013 and 2021, respectively. His research interests include natural language processing, machine learning, and artificial intelligence.

JONG-HYEOK LEE received the B.S. degree in mathematics education from Seoul National University, Seoul, South Korea, in 1980, and the M.S. and Ph.D. degrees in computer science from KAIST, Daejeon, South Korea, in 1982 and 1988, respectively. He is currently working as a Professor with the Graduate School of Artificial Intelligence, POSTECH, Pohang, South Korea. His research interests include natural language processing, machine translation, information retrieval, and artificial intelligence.