Computer Assisted Translation with Neural Quality Estimation and Automatic Post-Editing

Jiayi Wang[*], Ke Wang[*], Niyu Ge, Yangbing Shi, Yu Zhao, Kai Fan[†]
Alibaba Group Inc.
{joanne.wjy,moyu.wk,niyu.ge,taiwu.syb}@alibaba-inc.com,
kongyu@taobao.com, k.fan@alibaba-inc.com

Abstract

With the advent of neural machine translation, there has been a marked shift towards leveraging and consuming the machine translation results. However, the gap between machine translation systems and human translators needs to be manually closed by post-editing. In this paper, we propose an end-to-end deep learning framework of the quality estimation and automatic post-editing of the machine translation output. Our goal is to provide error correction suggestions and to further relieve the burden of human translators through an interpretable model. To imitate the behavior of human translators, we design three efficient delegation modules – quality estimation, generative post-editing, and atomic operation post-editing and construct a hierarchical model based on them. We examine this approach with the English–German dataset from WMT 2017 APE shared task and our experimental results can achieve the state-of-the-art performance. We also verify that the certified translators can significantly expedite their post-editing processing with our model in human evaluation.

1 Introduction

The explosive advances in the sequence to sequence model (Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017) enable the deep learning based neural machine translation (NMT) to approximate and even achieve the human parity in some specific language pairs and scenarios. Instead of translating from scratch by human translators, a new translation paradigm has emerged: computer assisted translation (CAT) system, which includes the machine translation and human post-editing. The post-editing is the process whereby humans amend machine-generated translations to achieve an acceptable final product. Practically, the estimated average translation time can be reduced by 17.4% (from 1957.4 to 1617.7 seconds per text) (Läubli et al., 2013).

However, utilizing NMT poses two key challenges. First, the neural machine translation quality still continues to vary a great deal across different domains or genres, more or less in proportion to the availability of paralleled training corpora. Second, the zero tolerance policy is a common choice in the vast majority of important applications. For example, when business legal documents are translated, even a single incorrect word could bring serious financial or property losses. Therefore, the subsequent human post-editing is indispensable in situations like this. Unfortunately, while NMT systems saves time by providing the preliminary translations, the time spent on error corrections by humans (Läubli et al., 2013) remains substantial to the extent that it offsets the efficiency gained by the NMT systems. In this paper, we explore automatic post-editing (APE) in the deep learning framework. Specifically, we adopt an imitation learning approach, where our model first screens the translation candidates by quality prediction and then decides whether to post edit with the generation or the atomic operation method.

Starting with a wide range of features used in the CAT system, we carefully analyze the human post-editing results to narrow down our framework design into three key modules: quality estimation (QE), generative post-editing and atomic operation post-editing. These modules are tightly integrated into the transformer neural networks (Vaswani et al., 2017). Our main innovation is a hierarchical model with two modular post-editing algorithms which are conditionally used based on a novel fine-grained quality estimation model. For each machine translation, our model i) runs the QE model to predict the detailed token level errors,

[*] indicates equal contribution.
[†] indicates corresponding author.

which will be further summarized as an overall quality score to decide whether the machine translation quality is high or not, and ii) conditional on the previous decision, employs the atomic operation post-editing algorithm on the high quality sentence or the generative model to rephrase the translation for the low one.

We examine our approach on the public English–German dataset from WMT[1] 2017 APE shared task. Our system outperforms the top ranked methods in both BLEU and TER metrics. In addition, following a standard human evaluation process aimed at achieving impartiality with respect to the efficiency of CAT system, we ask several certified translators to edit the machine translation outputs with or without our APE assistance. Evaluation results show that our system significantly improves translators' efficiency.

2 Related Work

Our work relates to and builds on several intertwined threads of research in machine translation, including QE and APE. We briefly survey the traditional methods and differentiate our approach.

2.1 Quality Estimation

Quality estimation is often a desired component for developing and deploying automatic language technologies, and has been extensively researched in machine translation (Barrault et al., 2019). Its purpose is to provide some metrics measuring the overall quality. The current state-of-the-art models mostly originated from the predictor-estimator framework (Kim et al., 2017), where a sequence-to-sequence model is pre-trained to extract sophisticated sequence features to be fed into a sequence level regression or classification network.

Tan et al. (2017) proposed the neural post-editing based quality estimation by streamlining together the traditional QE and APE models. Since our proposed QE module will eventually serve the APE module as well, we consider two modifications accordingly. First, we re-define the QE as a fine-grained multi-class problem, whose output indicates the number of tokens in four categories, missing / redundant / erroneous or kept tokens. A similar idea was initially proposed in (Gu et al., 2017) to predict the number of copy occurrences in non-autoregressive neural machine translation.

[1] http://www.statmt.org/

Table 1: Notation used in the model

Symbol	Definition
$\mathbf{s}$	sentence in source language
$\mathbf{m}$	machine translated sentence in target language
$\mathbf{t}$	golden (reference) sentence in target language
$\mathbf{e}$	post-editing sentence in target language
s_i	the i-th token of $\mathbf{s}$, similar for m_i, t_i, e_i
P_{MT}	the probabilistic model of machine translation
P_{PE}	the probabilistic model of post-editing
P_{QE}	the probabilistic model of quality estimation
$\mathbb{I}_A$	indicator function, $= 1$ if A is true, o.w. 0
τ	threshold to distinguish high/low quality translation

In this paper, we make significant extensions to include more categories. Secondly, we maximize our QE model performance with a novel conditional BERT architecture. Inspired by the masked language model objective in the encoder BERT (Devlin et al., 2019), we introduce the training objective to the encoder-decoder framework by adapting the decoder to become a memory encoder, allowing us to pre-train the target language model similar to BERT but conditioned on the source language text.

2.2 Automatic Post-Editing

Automatic Post Editing aims to improve the quality of an existing MT system by learning from human edited samples, converting "translationese" output into natural text. The traditional APE is based on a round-trip translation loop to mimic errors similar to the ones produced by NMT and can achieve acceptable performance with large scale monolingual data only (Freitag et al., 2019). However, the prevalent trend in this area prefers the dual-source encoder-decoder architecture with parallel data (Chatterjee et al., 2017b; Junczys-Dowmunt and Grundkiewicz, 2018; Pal et al., 2018; Lopes et al., 2019), which obtained the best results in WMT competitions (Chatterjee et al., 2019). The dual-source encoder encodes the source text and the machine translation output separately, and the decoder decodes the post-edited results. All these approaches encode each source independently and apply an auto-regressive decoder. They differ in their parameter sharing mechanisms.

While our approach still employs the multi-source APE framework, but there are two fundamental differences. First, our APE module, as aforementioned above, is built on our re-designed QE model, with which the source and the machine translation are entangled by the encoder and memory-encoder QE module. Second, our decoder

consists in a versatile architecture that can choose between the left to right auto-regressive generative model and the atomic-operation based paralleled model. It dynamically determines which model to engage at runtime. The parallelizable model was broadly explored in insertion- or deletion- based transformer (Chan et al., 2019; Stern et al., 2019; Gu et al., 2019), while our decoder supports more functional operations.

3 Model and Objective

In order to achieve the automatic post-editing goal, it is essential for the model to find the exact errors appearing in the machine translation and learn how to fix them. Breaking the problem into several sub-tasks, our proposed pipeline includes three major models as Figure 1. By skipping the pre-training temporarily, the first step is to investigate the fine-grained quality estimation model with respect to the source text and machine translated text. Its output will provide a fine-grained quality estimation of the machine translation. Based on the corresponding quality, an atomic APE or a generative APE model will be called for further processing.

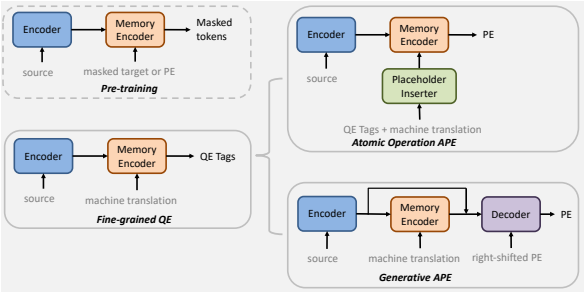

Figure 1: The overall pipeline. The QE model will output fine-grained metrics to the translation quality. Then, high quality machine translation will proceed with atomic APE model for minor fix, while the low quality machine translation will go through a generative APE model for completely rephrasing. Note that the model parameters are shared for three steps w.r.t. encoder and memory encoder. Detailed computational graph can refer to Figure 2.

3.1 Fine-Grained Quality Estimation

Table 2: Definition of QE Tags

Label	$k > 1$	$k = 1$	$k = 0$	$k = -1$
Definition	insert $k-1$ tokens	keep	delete	replace

As described in the related work, compared to traditional translation QE task in WMT[2], our QE module is more fine-grained and is recast as a multi-class $\{-1, 0, 1, ..., K\}$ sequence labeling problem. The definition of the integer labels is shown in Table 2. If $k <= 1$, the label denotes one single token operation; otherwise, it means to insert $k-1$ extra tokens after the current one. The QE tag $\mathbf{q}$ for training pair $(\mathbf{m}, \mathbf{e})$ can be deterministically calculated by dynamic programming Algorithm 4 in Appendix, which is basically a string matching algorithm. We define a conditionally independent sequence tagging model for the error prediction.

$$P_{\text{QE}}(\mathbf{q}|\mathbf{s}, \mathbf{m}) = \prod_i P_{\text{QE}}(q_i|\mathbf{s}, \mathbf{m}) \tag{1}$$

A transformer based neural network is employed. We present a novel encoder-memory encoder framework with memory attention as shown in the decomposition of the following equation.

$$\begin{aligned} & P_{\text{QE}}(\mathbf{q}|\mathbf{s}, \mathbf{m}) \\ \triangleq\ & \text{Softmax}_{\text{QE}}(\text{Enc}^M(\mathbf{m}, \text{Enc}(\mathbf{s}))) \end{aligned} \tag{2}$$

where $\text{Enc}(\cdot)$ is the standard transformer encoder (Vaswani et al., 2017), and $\text{Enc}^M(\cdot)$ is the memory encoder adapted from standard transformer decoder. It removed the future masking in the transformer decoder and use the last state as the output which contains contexts from both SRC and MT.

During inference, neither the ground truth of post-editing nor the golden translation reference is available. The fine-grained QE model can predict the human translation edit rate (HTER) h through the inferred QE tags $\hat{\mathbf{q}}$.

$$\begin{aligned} h &= \frac{\#\text{predicted edits}}{\text{predicted PE length}} \\ &= \frac{\sum_i \{\mathbb{I}_{\hat{q}_i<1} + (\hat{q}_i - 1)\mathbb{I}_{\hat{q}_i>=1}\}}{\sum_i |\hat{q}_i|} \end{aligned} \tag{3}$$

On the one hand, the overall metric h can quantitate the quality of machine translation and determine which APE algorithm will be used. On the other hand, the detailed QE tags can theoretically guide the APE which atomic operation should be applied. Thus, the QE tagging and the atomic operation APE are simultaneously and iteratively trained, which will be elaborated in 3.2 and 3.5.

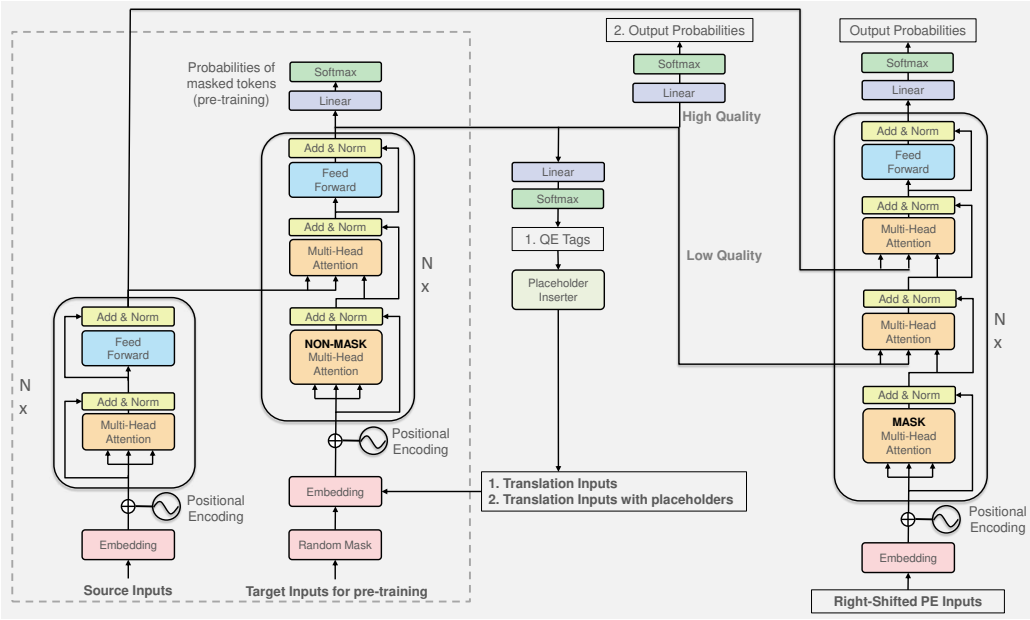

Figure 2: The detailed computational graph including detailed operations.

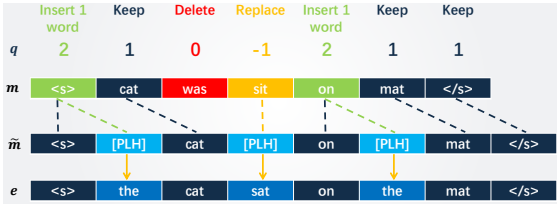

Figure 3: An example illustration of placeholder inserter and atomic operation APE.

3.2 Atomic Operation Automatic Post-Editing

The key idea of atomic operation APE is to reduce all predefined operations (insertion, deletion, substitution) into a special substitution operation by introducing an artificial token placeholder [PLH].

First, we align the machine translation $\mathbf{m}$ and the post-edits $\mathbf{e}$ by inserting [PLH]s, resulting in a new $\tilde{\mathbf{m}}$ of the same length as $\mathbf{e}$. Technically, we insert $q_i - 1$ [PLH]s after m_i if $q_i > 1$; we delete the current token m_i if $q_i = 0$; we replace m_i with [PLH] if $q_i = -1$. For convenience, this process is denoted as $\tilde{\mathbf{m}} = \text{PLH\_INS}(\mathbf{m}, \mathbf{q})$.

Second, the original APE task is transformed into another sequence tagging problem, since $|\tilde{\mathbf{m}}| = |\mathbf{e}|$.

$$\begin{aligned} P^A_{\text{PE}}(\mathbf{e}|\mathbf{s},\mathbf{m}) &= P^A_{\text{PE}}(\mathbf{e}|\mathbf{s},\tilde{\mathbf{m}}) \\ &= \text{Softmax}_{\text{PE}}(\text{Enc}^M(\tilde{\mathbf{m}}, \text{Enc}(\mathbf{s}))) \end{aligned} \quad (4)$$

[2] http://www.statmt.org/wmt19/qe-task.html

Notice that **i)** the encoder and memory encoder share the parameters with the QE in Equation (2); **ii)** the softmax layer is different, because the number of outputs in APE has a different size equal to the vocabulary size. An intuitive visualization can see the Figure 3 and the holistic pipeline sees the Figure 1.

3.3 Generative Automatic Post-Editing

The larger HTER h is, the lower quality of $\mathbf{m}$ is, and the more atomic operations are required. In this case, the previous APE model may be not powerful enough to learn complicated editing behaviors. We propose a backup APE model via auto-regressive approach for the deteriorated translations. Concretely, we write the dual-source language model into its probabilistic formulation.

$$\begin{aligned} P^G_{\text{PE}}(\mathbf{e}|\mathbf{s},\mathbf{m}) &= \prod_i P^G_{\text{PE}}(e_i|\mathbf{e}_{<i},\mathbf{s},\mathbf{m}) \\ &= \prod_i \text{Dec}(\mathbf{e}_{<i}; \text{Enc}^M(\mathbf{m}, \text{Enc}(\mathbf{s})); \text{Enc}(\mathbf{s})) \end{aligned} \quad (5)$$

Notice that **i)** the encoder and memory encoder are still reused here, **ii)** the $\text{Dec}(\cdot;\cdot;\cdot)$ is a transformer decoder with hierarchical attention, since two memory blocks $\text{Enc}^M(\mathbf{m}, \text{Enc}(\mathbf{s}))$ and $\text{Enc}(\mathbf{s})$ are both conditional variables for the auto-regressive language model; **iii)** unlike sequence tagging, the inference of the generative APE is intrinsically non-parallelizable.

Algorithm 1 Imitation Learning Algorithm

Require: $\mathbf{s}$, $\mathbf{m} = \{m_i\}_{i=1}^{M}$, $\mathbf{e} = \{e_i\}_{i=1}^{N}$, hyperparameter $\beta \in (0, 1)$.

Draw a random number r from uniform distribution $[0, 1]$.
if $r > \beta$ **then**
  $\tilde{\mathbf{m}} = \text{PLH\_INS}(\mathbf{m}, \mathbf{q})$.
else
  Randomly replace 20% of e_i as [PLH] to obtain $\tilde{\mathbf{m}}$.
end if
Pseudo data for insertion Remove all [PLH] in $\tilde{\mathbf{m}}$ to obtain $\mathbf{m}^i$.
Pseudo data for substitution Run APE inference model to obtain the prediction $\hat{\mathbf{e}}^s \leftarrow P_{\text{PE}}^A(\cdot|\mathbf{s}, \tilde{\mathbf{m}})$.
Pseudo data for deletion Randomly insert one or two [PLH]s to each gap in $\mathbf{e}$ with probability 0.15 or 0.025 to obtain the updated $\tilde{\mathbf{m}}$.
Run APE inference model to obtain the prediction $\hat{\mathbf{e}}^d \leftarrow P_{\text{PE}}^A(\cdot|\mathbf{s}, \tilde{\mathbf{m}})$.
return 3 fake data points, $\mathbf{m}^i, \mathbf{m}^s = \hat{\mathbf{e}}^s, \mathbf{m}^d = \hat{\mathbf{e}}^d$.

3.4 Pre-training and Imitation Learning

Because of the lack of human post-editing data, training from scratch is typically difficult. We thus employ two workaround methods to improve the model performance.

Pre-training It is worth noting that the reduced atomic operation APE is actually equivalent to the mask language modeling problem, a.k.a. the famous BERT (Devlin et al., 2019). Therefore, we pre-train the encoder-memory encoder model as a conditional BERT with the data pairs $(\mathbf{s}, \mathbf{t})$ and $(\mathbf{m}, \hat{\mathbf{e}})$, aiming at learning the syntactic and alignment information of the ground truth. To make the pre-training valid on downstream tasks, we consistently use [PLH] token to randomly mask the reference / post-editing sentence.

Imitation Learning As mentioned in 3.1, during inference, the predicted QE tags will causally tie to the successive APE algorithm, because $\tilde{\mathbf{m}}$ is derived from $(\mathbf{m}, \hat{\mathbf{q}})$. Although we would want the model to learn to predict all three atomic operations together, the small size of real post-editing data severely limits the performance of joint QE tagging. Therefore, we propose a model specialization strategy where the model learns three separate tasks: deletion, insertion, and substitution. A reasonable amount of training data can be generated for each of the tasks and the model learns to specialize in each operation. The details are summarized in Algorithm 1.

3.5 Training and Inference Algorithms

In this section, we assemble all modules together into the final system. Because our model involves a nontrivial pipeline, we describe the details of training and inference separately and summarize them in Algorithm 2 and 3.

Algorithm 2 APE Training

Require: Pre-training data $\mathcal{P}$ in pair ($\mathbf{s}$, $\mathbf{t}$ or $\mathbf{e}$), QE Training data $\mathcal{Q}$ in triplet$(\mathbf{s}, \mathbf{m}, \mathbf{e})$.

Pre-train the encoder-memory encoder model with $\mathcal{P}$ as 3.4.
while not converge **do**
  Sample a tuple from $\mathcal{Q}$.
  Call Algorithm 1 to enlarge the training sample four times.
  for each $(\mathbf{s}, \mathbf{m}, \mathbf{e})$ in the augmented data **do**
    Calculate true QE tags $\mathbf{q}$ =Algorithm 4$(\mathbf{m}, \mathbf{e})$.
    Get machine translation with [PLH] $\tilde{\mathbf{m}} = \text{PLH\_INS}(\mathbf{m}, \mathbf{q})$.
    Update model parameters of encoder-memory encoder by optimizing the loss $\mathcal{L}_{\text{QE}}(\mathbf{q}, \mathbf{s}, \mathbf{m}) + \mathcal{L}_{\text{PE}}^A(\mathbf{e}, \mathbf{s}, \tilde{\mathbf{m}})$.
    Update All model parameters by optimizing loss $\mathcal{L}_{\text{PE}}^G(\mathbf{e}, \mathbf{s}, \mathbf{m})$.
  end for
end while
return All model parameters.

Algorithm 3 APE inference

Require: $\mathbf{s}$, $\mathbf{m}$, HTER threshold τ, iteration steps S.

$\mathbf{m}^{(0)} = \mathbf{m}$
for $i = 1, ..., S$ **do**
  Run QE inference $\hat{\mathbf{q}} \leftarrow P_{\text{QE}}(\cdot|\mathbf{s}, \mathbf{m}^{(i-1)})$.
  Run Equation 3 to obtain quality metric h.
  if $i == 1$ and $h > \tau$ **then**
    Run generative APE inference $\hat{\mathbf{e}} \leftarrow P_{\text{PE}}^G(\cdot|\mathbf{s}, \mathbf{m})$.
    return APE $\hat{\mathbf{e}}$.
  end if
  $\tilde{\mathbf{m}} = \text{PLH\_INS}(\mathbf{m}^{(i-1)}, \mathbf{q})$
  Run atomic operation APE inference $\mathbf{m}^{(i)} \leftarrow P_{\text{PE}}^A(\cdot|\mathbf{s}, \tilde{\mathbf{m}})$.
end for
return APE $\hat{\mathbf{e}} = \mathbf{m}^{(S)}$.

Training usually requires to minimize the loss function (negative data log-likelihood of probabilistic models) by stochastic gradient descent (SGD) with respect to the trainable parameters. Our QE and atomic operation APE are both sequence tagging task, while the generative APE is a sequence generation task. The three loss functions are uniformly defined as sequential cross entropy between the predicted and the true sequence. Note that the QE and atomic operation APE share the encoder-memory encoder, so these two losses can be summed together for optimization. However, the generative APE model has an isolated hierarchical transformer decoder, so we need a second update by optimizing the corresponding loss alone.

Inference of our APE system is not quite the same as the training. First, the overall inference is a continuously alternating procedure between QE

and APE, where the predicted APE is assigned as a new machine translation for iterative updating. However, the inner loop in training algorithm regards to the augmented data points. Second, we introduce an early stop after the first QE tagging prediction. If the predicted quality is very low (*i.e.* the HTER is larger than a cross-validated threshold), the generative APE will be called and the inference will immediately exit without further iterations. Lastly, the APE results are utilized by professional translators for further editing. In the next section, we validate the gain of APE over machine translation with regards to the efficiency.

4 Experiments on our Proposed Model

We verify the validity and efficiency of the proposed APE model by conducting a series of APE experiments and human evaluation on WMT'17 APE dataset. For convenience, we denote the generative post-editing model as *GM*, the atomic operation post-editing model as *AOM*, and the final hierarchical model as *HM* in this section.

4.1 Setup

Dataset. The open public WMT17 Automatic Post-Editing Shared Task (Bojar et al., 2017) data on English-German (En-De) is widely used for APE experiments. It consists of 23K real triples (source, machine translation & post-editing) for training and another 2K triples for testing from the Information Technology (IT) domain. Besides, the shared task also provides a large-scale artificial synthetic corpus containing around 500K high quality and 4 million low quality synthetic triples. We over sample the APE real data by 20 times and merge it with the synthetic data, results in roughly 5 million of triples for both pre-training and APE training. The details of the training set are shown in Appendix Table 6. We adopt test set of the same task in WMT16 as the development set. Furthermore, we apply truecaser (Koehn et al., 2007) to all files and encode every sentence into subword units (Kudo, 2018) with a 32K shared vocabulary.

Evaluation Metrics. We mainly evaluate our systems with metrics bilingual evaluation understudy (**BLEU**) (Papineni et al., 2002) and translation edit rate (**TER**) (Snover et al., 2006), since they are standard and widely employed in the APE shared task. The metric BLEU indicates how similar the candidate texts are to the reference texts, with values closer to 100 representing higher similarity. TER measures how many edits required from the predicted sentence to the ground truth sentence, and is calculated by Equation (3) as well and multiplied by 100.

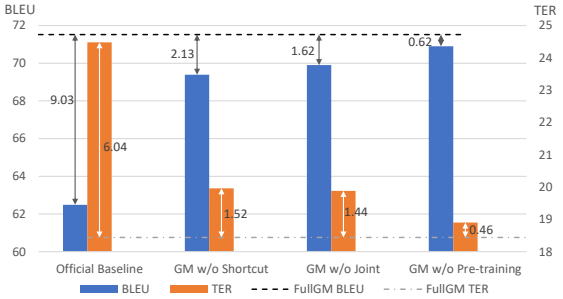

Figure 4: Results of Our Generative Model on Test Set

Training Details. All experiments are trained on 8 NVIDIA P100 GPUs for maximum 100,000 steps for about two days until convergence, with a total batch-size of around 17,000 tokens per step and the Adam optimizer (Kingma and Ba, 2014). Only the source and post-edited sentence pairs are used for pre-training. During pre-training, 20% tokens in post-editing sentence are masked as [PLH]. Parameters are being tuned with 12,000 steps of learning rates warm-up (Vaswani et al., 2017) for both of the GM and AOM model. However, 5 automatic post editing iterations (*i.e.* $S = 5$ in Algorithm alg:infer) are applied during the inference for the AOM model due to its characteristic of fine-grained editing behaviors. Except these modifications, we follow the default transformer-based configuration (Vaswani et al., 2017) for other hyper-parameters in our models.

4.2 APE Systems Comparison

The main results of automatic post-editing systems are presented in Table 3 and competitively compared with results of recent years' winners of WMT APE shared task and several other top results. It is observed that our hierarchical single model achieves the state-of-the-art performance on both BLEU and TER metrics, outperforming not only all other single models but also the ensemble models of top ranked systems in WMT APE tasks.

Note that our hierarchical system is not a two-model ensemble. The standard ensemble method requires inference and combination of results from more than one models. In contrast, our hierarchical model contains multiple parameter-sharing modules to accomplish multi-tasks, and only need to infer once on the selected model.

Table 3: Performance Comparison on WMT17 APE En-De Dataset

Model	BLEU↑	TER↓	Note
Official Baseline	62.49	24.48	Do nothing with the origin machine translation
MS-UEdin	69.72	19.49	Single model (Junczys-Dowmunt and Grundkiewicz, 2018), winner of WMT18 APE task
Levenshtein Transformer	70.1	19.2	Single model (Gu et al., 2019)
Unbabel	70.66	19.03	Single model (Correia and Martins, 2019), winner of WMT19 APE task.
FBK (Ensemble)	70.07	19.60	Ensemble model(Chatterjee et al., 2017a), winner of WMT17 APE task
MS-UEdin (Ensemble)	70.46	19.03	Ensemble model(Junczys-Dowmunt and Grundkiewicz, 2018)
Unbabel (Ensemble)	71.90	18.07	Ensemble model(Correia and Martins, 2019)
Only GM	71.52	18.44	Single model, *i.e.* $\tau = 0$ in Algorithm 3
Only AOM	68.40	20.34	Single model, *i.e.* $\tau = 1$ in Algorithm 3
Our HM	**72.07**	**18.01**	Single model, *i.e.* $\tau = 0.3$, determined on development dataset

Table 4: Performance Gain from Pseudo Data

Model	BLEU↑	TER↓	ΔBLEU	ΔTER
AOM w/o pseudo data	65.65	22.14	-	-
AOM with pseudo data	68.40	20.34	+**2.75**	−**1.80**

4.2.1 Results of Generative APE Model

As mentioned in section 3.3, the decoder of our generative model receives encoder-memory encoder outputs, refering to SRC memory and SRC-MT joint memory. A transformer attention layer encodes the SRC into the SRC memory, and the joint memory is produced by another one, which encodes the original MT conditionally on the SRC memory. These two encoders are pre-trained with sources and post-edits from the full training data.

We designed a set of systematic experiments to verify that our model benefits from such a design in Figure 4: (1) To verify that the memory encoder has the ability to learn cross-lingual knowledge, we replace the memory encoder with an ordinary multi-head self-attention encoder, which does not accept the source memory as input, marked by *w/o Joint*. (2) To prove that the shortcut from the SRC memory to the decoder input is necessary, the shortcut is removed in the *w/o Shortcut* experiment. (3) To verify that our model can leverage representations from pre-training, we conduct an experiment without pre-training, denoted as *w/o Pre-training*.

The ablation results significantly demonstrate that our model does benefit from meory encoder, SRC memory shortcut and pre-training. Removing any of them will result in performance loss.

4.2.2 Results of Atomic Operation APE Model

In each iteration, based on the QE model's output, our AOM refines the MT in parallel regarding to all placeholders. Unlike the GM, the time cost of the AOM only depends on the steps of iterations, regardless of the length of the sentence. To evaluate the decoding efficiency, we collect the AOM's performances at different iteration steps, as shown in Figure 5.

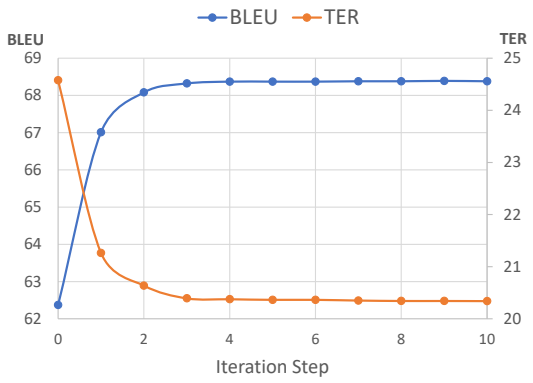

Figure 5: The convergence curves of the AOM inference w.r.t. iteration. The iterative updating converges within only 3 to 5 steps, which is much smaller than the averaged number of decoding steps of the GM.

The Role of Pseudo Data. As noted in section 3.4, model specialization algorithm is applied to train the model to learn different kinds of atomic operations. We compare our AOM on the test set with and without pseudo data in Table 4. The results demostrate that our model specialization algorithm plays a key role by providing a powerful guidance for training and making up for the deficiency from the lack of large amount of real APE data.

4.2.3 Results of QE Model

The QE model is the prerequisite of the final hierarchical model as well as the basis of our atomic operation model. Therefore, it is necessary to guarantee the performance of QE results as accurate as possible. Unlike the traditional OK/BAD word-

Table 5: Results of Fine-Grained QE Model (Pearson = 0.664). Quality tag prediction is evaluated in terms of multi-classification accuracy via F1-scores. The overall MT quality estimation is measured by the Pearson correlation coefficient, indicating the correlation between the predicted and the real MT quality w.r.t. TER.

	K	E	R	M	OK	BAD
Precision↑	0.877	0.710	0.563	0.622	0.898	0.783
Recall↑	0.951	0.471	0.480	0.540	0.962	0.559
F1-score↑	0.913	0.566	0.518	0.578	0.928	0.652

level QE task in WMT (Bojar et al., 2017), our model pursues to predict fine-grained quality tags. So, we cannot make a completely fair comparison with previous works.

The fine-grained quality tag of each word predicted by the model can be classified into one of the four labels: *K* for **Kept**, *E* for **Erroneous** , *R* for **Redundant** and *M* for **Missing**. Furthermore, we convert the predicted fine-grained QE tags to OK/BAD tags directly by treating tag *K* and tag *M* as *OK*, and the other two tags as *BAD* according to the rules of tagging in WMT17 QE Shared Task.

We provide our fine-grained QE results on the test dataset of WMT17 APE Task in Table 5, where the ground-truth tags are produced by Algorithm 4 in Appendix A.1. Note that the TER score can be easily computed from the predicted quality tags. The predicted TER score is regarded as an indicator of MT quality in our hierarchical model: MTs with quality higher than τ in Algorithm 3 are fed to the GM, otherwise they are sent to the AOM. The hyper-parameter $\tau = 0.3$ is determined by cross validation on WMT16 development dataset. Afterwards, we apply it on the WMT17 test dataset to select a potentially preferable model from GM and AOM to generate the final APE result for each SRC and MT pair.

There are more than 75% of tokens in the training set are tagged with *Keep*. In terms of the huge challenge posed by the unbalanced dataset, our fine-grained quality estimation is quite remarkable. The performance of our final hierarchical model in Table 3 proves the effectiveness of it.

4.3 Results of Human Evaluation

We conduct real post-editing experiments with professional translators involved. There are 6 independent participating translators, randomly divided into 2 groups. They are all native speakers of German and have 10+ years of experience in translation of En-De in IT related domains. We follow two different flows in our experiments. For fair comparison, both of the two groups see the same 100 source sentences picked from the WMT17 test dataset. The MTs are provided for the first group for post-editing, while our model generated APEs for the second group. However, the information on the category of the translation is not revealed to translators. The translators are asked to record the elapsed time of their labor in total.

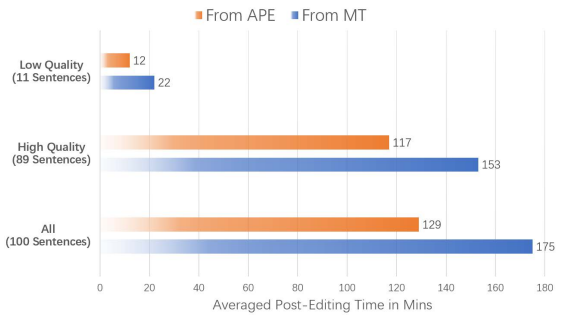

Figure 6: Time Spent in Post-Editing by Translators. The averaged total time spent by translators to post-edit the APE becomes significantly decreased by 26.3%

The statistics of averaged post-editing time for different translators are summarized in Figure 6. Besides the total time, we also analyze the duration for low and high quality translations separately (determined by QE model). In either case, post-editing from the APE costs less time. We also did case study about high-quality vs low-quality APE in Appendix A.3. From different perspectives of experimental validation, we can conclude that the APE generated by our model can ease the burden of translators and substantially improve the post-editing efficiency.

5 Conclusion

In this paper, we proposed a hierarchical model that utilizes the fine-grained word-level QE prediction to select one of the two APE models we proposed to generate better translations automatically, which shows a state-of-the-art performance. In particular, we designed a dynamic deep learning model using imitation learning, which intuitively mimics the editing behaviors of human translators. Our hierarchical model is not a standard ensemble model in the conventional sense. We merely shared the parameters of different modules to accomplish different objectives, including QE, AOM and GM. Our experimental findings show that if the characteristics of errors in the machine translation can be

accurately simulated, it is highly likely that MT output can be automatically refined by the APE model. Towards this end, we conduct a rigorous comparison of the machine translation and automatic post-editing based manual post-editing tasks, and it is observed that the latter can significantly increase the efficiency of post-editing.

Acknowledgments

This work is partly supported by National Key R&D Program of China (2018YFB1403202).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

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

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. Kermit: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*.

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

Rajen Chatterjee, M Amin Farajian, Matteo Negri, Marco Turchi, Ankit Srivastava, and Santanu Pal. 2017b. 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.

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

Gonçalo M. Correia and André F. T. Martins. 2019. A simple and effective approach to automatic post-editing with transfer learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3050–3056, Florence, Italy. Association for Computational Linguistics.

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

Markus Freitag, Isaac Caswell, and Scott Roy. 2019. Ape at scale and its implications on mt evaluation biases. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 34–44.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2018. Ms-uedin submission to the wmt2018 ape shared task: Dual-source transformer for automatic post-editing. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 835–839. Association for Computational Linguistics.

Hyun Kim, Hun-Young Jung, Hongseok Kwon, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-estimator: Neural quality estimation based on target word prediction for machine translation. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 17(1):3.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. pages 66–75.

Samuel Läubli, Mark Fishel, Gary Massey, Maureen Ehrensberger-Dow, Martin Volk, Sharon O'Brien, Michel Simard, and Lucia Specia. 2013. Assessing post-editing efficiency in a realistic translation environment.

António V. Lopes, M. Amin Farajian, Gonçalo M. Correia, Jonay Trénous, and André F. T. Martins. 2019. Unbabel's submission to the WMT2019 APE shared task: BERT-based encoder-decoder for automatic post-editing. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 118–123, Florence, Italy. Association for Computational Linguistics.

Santanu Pal, Nico Herbig, Antonio Krüger, and Josef van Genabith. 2018. A transformer-based multi-source automatic post-editing system. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 827–835.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc Meeting of the Association for Computational Linguistics*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yiming Tan, Zhiming Chen, Liu Huang, Lilin Zhang, Maoxi Li, and Mingwen Wang. 2017. Neural post-editing based on quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 655–660.

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

A Appendix

A.1 Pseudo code of QE tag computation

The computation of QE tags is quite similar to the famous Minimum Edit Distance problem and can be solved with dynamic programming in algorithm 4.

Algorithm 4 QE tag computation

Require: machine translation $\mathbf{m} = \{m_i\}_{i=1}^{M}$, post-editing $\mathbf{e} = \{e_i\}_{i=1}^{N}$.

Initialize the edit distance matrix $d_{i,0} = i, d_{0,j} = j$ and QE tag $q_i = 1$.
for $i = 1, ..., M$ **do**
  for $j = 1, ..., N$ **do**
    $d_{i,j} = \min\{d_{i-1,j-1} + \mathbb{I}_{m_i \neq e_j}, d_{i,j-1} + 1, d_{i-1,j} + 1\}$
  end for
end for
while $i > 0$ or $j > 0$ **do**
  if $i > 0$ and $j > 0$ and $d_{i-1,j-1} + 1 = d_{i,j}$ **then**
    $q_i = -1, i--, j--$
  else if $j > 0$ and $d_{i,j-1} + 1 = d_{i,j}$ **then**
    $q_i++, j--$
  else if $i > 0$ and $d_{i-1,j} + 1 = d_{i,j}$ **then**
    $q_i = 0, i--$
  else
    $i--, j--$
  end if
end while
return $\mathbf{q} = \{q_i\}_{i=1}^{M}$

A.2 Details of the Traning Corpus

WMT APE shared-task provided both real APE triplets and a large a large-scale artificial synthetic corpus containing around 500K high quality and 4 million low quality synthetic triples. Table 6 shows the difference between them.

Table 6: Details of the WMT 2017 APE Shared-Task Dataset. The BLEU and TER metrics are directly evaluated on machine translation and post-editings as references.

Source	# Sentence	Avg. Length	BLEU	TER
Real Triples	23,000	17.88	61.87	25.35
Artificial 500K	526,368	20.90	60.01	25.55
Artificial 4M	4,391,180	16.68	46.59	35.37
500K+20∗Real	986,368	19.49	60.80	25.46
4M+500K+20∗Real (Full Training data)	5,377,548	17.20	49.65	33.31

A.3 Case Study and Runtime Efficiency

As mentioned in the paper, the AOM is more suitable for translations that only require a few edit operations while GM is more preferable for low quality translations. To demonstrate this conclusion and prove the effectiveness of our QE-based automatic selector, some cases of translations with different qualities are shown in Table 7.

In case 1 and case 2, the translation is quite close to *pe*. Therefore, the AOM only need to predict tokens for a small number of [PLH]s. When there are relatively complete contexts provided, the AOM can achieve a higher performance than the GM. Moreover, after reading the source and the final output, the human translators did not even take any additional action to improve the translation quality.

In the opposite way, as shown in case 3 and case 4, there is a huge gap between *mt* and *pe*, and the input for AOM contains a considerable number of placeholders, which lacks enough contextual information. In these cases, our GM can auto-regressively regenerate the translation based on the given *mt* to guarantee the higher quality of the final output. Based on the QE selector, the translators only need to make very few efforts to correct the errors in the final generated APE of our model.

A practical point of the computer assisted translation via APE is its expense and computational cost. Compared with the traditional computer assisted translation crowdsourcing, machine translation + human post-editing, our additional automatic post-editing does increase the computational cost, which is roughly equivalent to another machine translation model. In general, the crowdsourcing is charged by hours. The numbers in our findings suggest a promising budget cut associated with CAT crowdsourcing. However, this extra APE module may lead to a latency increase by $\tilde{4}$00ms, which is still far below the average time cost by human post-editing. Even for an online crowdsourcing system, a well-designed concurrent mechanism should make the translators not feel any delay. From the perspective of architecture scale, the APE model can be deployed in the identical processing unit for the machine translation model and be called successively in a pipeline. The only concern is that the memory storage capacity should be large enough to store more parameters.

Table 7: Examples of Crowdsourcing after APE. Tokens in "⟨⟩" indicates GM's over corrections or AOM's inaccurate translations due to too many consecutive [PLH] predictions, which leads inadequate contextual information. Tokens in "{}" highlights correct automatic editings.

	High Quality s Translation Case	
	SRC	In List view , click any column header to sort by that criteria .
	MT	Klicken Sie in der Listenansicht auf eine beliebige Spaltenberschrift , um nach dieser Kriterien sortieren .
	PE	Klicken Sie in der Listenansicht auf eine beliebige Spaltenberschrift , um nach diesen Kriterien zu sortieren .
	MT (sub-word)	_klicken _Sie _in _der _Listenansicht _auf _eine _beliebige _Spalten berschrift _, _um _nach _dieser _Kriterien _sortieren _.
Case1	Predicted QE Tag	1 1 1 1 1 1 1 1 1 1 1 1 -1 2 1 1
	TER vs Predicted TER	11.76 vs 11.11
	AOM Input	_klicken _Sie _in _der _Listenansicht _auf _eine _beliebige _Spalten berschrift _, _um _nach [PLH] _Kriterien [PLH] _sortieren _.
	AOM Output	_klicken _Sie _in _der _Listenansicht _auf _eine _beliebige _Spalten berschrift _, _um _nach {_**diesen**} _Kriterien {_**zu**} _sortieren _.
	GM Output	_klicken _Sie _in _der _Listenansicht _auf _eine _beliebige _Spalten berschrift _, _um _nach _dieser _Kriterien {_**zu**} _sortieren _.
	Final Output	Klicken Sie in der Listenansicht auf eine beliebige Spaltenberschrift , um nach **diesen** Kriterien **zu** sortieren .
	Translator Edit	no action
	SRC	You can justify all text in a paragraph either including or excluding the last line .
	MT	Sie knnen den gesamten Text eines Absatzes mit oder ohne die letzte Zeile .
	PE	Sie knnen den gesamten Text eines Absatzes mit oder ohne die letzte Zeile ausrichten .
	MT (sub-word)	_Sie _knnen _den _gesamten _Text _eines _Absatzes _mit _oder _ohne _die _letzte _Zeile _.
	Predicted QE Tag	1 1 1 1 1 1 1 1 1 1 1 1 2 1
Case2	TER vs Predicted TER	6.67 vs 6.67
	AOM Input	_Sie _knnen _den _gesamten _Text _eines _Absatzes _mit _oder _ohne _die _letzte _Zeile [PLH] _.
	AOM Output	_Sie _knnen _den _gesamten _Text _eines _Absatzes _mit _oder _ohne _die _letzte _Zeile {_**ausrichten**} _.
	GM Output	_Sie _knnen _den _gesamten _Text _eines _Absatzes ⟨_**entweder** _**einschlielich**⟩ _oder _ohne _die _letzte _Zeile _lschen _.
	Final Output	Sie knnen den gesamten Text eines Absatzes mit oder ohne die letzte Zeile **ausrichten** .
	Translator Edit	no action
	Low Quality Translation Case	
	SRC	In Start Number , enter the number to assign to the first PDF on the list .
	MT	Whlen Sie unter " Number , " geben Sie die Nummer fr die erste PDF-Datei in der Liste aus .
	PE	Geben Sie unter " Startnummer " die Nummer fr die erste PDF-Datei in der Liste ein .
	MT (sub-word)	_whlen _Sie _unter _" _Number _, _" _geben _Sie _die _Nummer _fr _die _erste _PDF - Datei _in _der _Liste _aus _.
Case3	Predicted QE Tag	-1 1 1 2 -1 -1 -1 -1 -1 1 1 0 -1 -1 1 1 1 -1 1 1 -1 1
	TER vs Predicted TER	35.29 vs 54.55
	AOM Input	[PLH] _Sie _unter _" [PLH] [PLH] [PLH] [PLH] [PLH] [PLH] _die _Nummer [PLH] [PLH] _PDF - Datei [PLH] _der _Liste [PLH] _.
	AOM Output	{_**geben**} _Sie _unter _" _Start ⟨_**geben** _**Sie** _**zum** _**Zuweisen** _⟩" _die _Nummer _der _ersten _PDF - Datei _ber _der _Liste {_**ein**} _.
	GM Output	{_**geben**} _Sie _unter _" {_**Start nummer**} _" _die _Nummer _fr _die _erste _PDF - Datei _in _der _Liste _an _.
	Final Output	**Geben** Sie unter " **Startnummer** " die Nummer fr die erste PDF-Datei in der Liste an .
	Translator Edit	an→ein
	SRC	The Illustrator text is converted to HTML text with basic formatting attributes in the resulting web page .
	MT	Die Illustrator Text HTML-Text mit grundlegenden Formatierungsattribute in der erstellten Webseite konvertiert wird .
	PE	Die Illustrator-Text wird in HTML-Text mit grundlegenden Formatierungsattributen in der erstellten Webseite konvertiert .
	MT (sub-word)	_die _Illustrator _Text _HTML - Text _mit _grundlegenden _Formatierung s attribute _in _der _erstellten _Webseite _konvertiert _wird _.
Case4	Predicted QE Tag	-1 3 3 1 1 1 1 1 1 1 -1 1 1 1 1 1 0 1
	TER vs Predicted TER	35.29 vs 33.33
	AOM Input	[PLH] _Illustrator [PLH] [PLH] _Text [PLH] [PLH] _HTML - Text _mit _grundlegenden _Formatierung s [PLH] _in _der _erstellten _Webseite _konvertiert _.
	AOM Output	_in _Illustrator - Der _Text _in _in _HTML - Text _mit _grundlegenden _Formatierung s {**attributen**} _in _der _erstellten _Webseite _konvertiert _.
	GM Output	_der _Illustrator {**- Text** _**wird** _**in**} _HTML - Text _mit _grundlegenden _Formatierung s {**attributen**} _in _der _erstellten _Webseite _konvertiert _.
	Final Output	Der Illustrator-**Text wird in** HTML-Text mit grundlegenden **Formatierungsattributen** in der erstellten Webseite konvertiert .
	Translator Edit	Der→Die