NiuTrans Submission for CCMT19 Quality Estimation Task

Ziyang Wang[1], Hui Liu[1], Hexuan Chen[1], Kai Feng[1], Zeyang Wang[1], Bei Li[1], Chen Xu[1], Tong Xiao[1,2(✉)], and Jingbo Zhu[1,2]

[1] NLP Lab, Northeastern University, Shenyang, China
{wangziyang,huiliu,chenhexuan,fengkai,wangzeyang,
libeinlp,xuchen}@stumail.neu.edu.cn
[2] NiuTrans Co., Ltd., Shenyang, China
{xiaotong,zhujingbo}@mail.neu.edu.cn

Abstract. This paper describes our system submitted for the CCMT 2019 Quality Estimation (QE) Task, including sentence-level and word-level. We propose a new method based on predictor-estimator architecture [7] in this task. For the predictor, we adopt Transformer-DLCL [17] (dynamic linear combination of previous layers) as our feature extracting models. In order to obtain the information of translations in both directions, we use right-to-left and left-to-right two models, concatenate two feature vectors as whole quality feature vectors. For the estimator, we use a multi-layer bi-directional GRU to predict `HTER` scores or `OK/BAD` labels for different tasks. We pre-train the predictor according to machine translation (MT) method with bilingual data from WMT2019 EN-ZH task, and then jointly train predictor and estimator with the QE task data. We also construct 50K pseudo data in different methods in respond to the data scarcity. The final system integrates multiple single models to generate results.

Keywords: Quality estimation · Deep Transformer · Bi-GRU

1 Introduction

Quality estimation (QE) refers to the task of evaluating the quality of MT results without any human annotated references [2]. We participate the CCMT 2019 QE task in both EN→ZH and ZH→EN directions. Each of them consists of two subtasks: word-level and sentence-level. Word level task is to predict `OK/BAD` labels for each word and gap in translation results, corresponding to mistranslation, over-translation and under-translation. Sentence-level task is to predict the Human-targeted Translation Edit Rate (`HTER`) scores [14] which represent the overall quality of the translation results.

In early works, human-crafted features were wildly used. A typical framework was QUEST++ [15] which provided a variety of features and machine learning methods to build QE models. In recent years, neural models significantly improved the performance in this task. Kim et al. [7] proposed a neural

S. Huang and K. Knight (Eds.): CCMT 2019, CCIS 1104, pp. 82–92, 2019.
https://doi.org/10.1007/978-981-15-1721-1_9

network architecture called predictor-estimator, which adopted a bilingual recurrent neural network (RNN) language model [9] as predictor to extract feature vectors, and used a bidirectional RNN as estimator to predict QE scores. Fan et al. [5] introduced a bidirectional Transformer based pre-trained model for feature extraction, and used 4-dimensional mis-matching features from this model to improve performance.

In our work, all the tasks we submit share the same model architecture based on the predictor-estimator. We pre-train left-to-right and right-to-left deep Transformer models with a large amount of bilingual data as predictor. Byte-pair-encoding (BPE) [12] tokenization is applied to reduce the number of unknown tokens. After that, a multi-layer Bi-GRU is used as estimator, and is jointly trained with predictors using the quality estimation task data. We transform word-level tasks into binary classification problems and sentence-level tasks into regression problems for estimator model to predict labels or scores with the feature information extracted by predictor.

To further improve the performance of the predictor, we use target-side monolingual data to construct pseudo-data by various back-translation [3] methods, including beam search, sampling and sampling-topk [4]. Due to the scarcity of QE data, we also construct QE pseudo data. We regard real target-side sentences in bilingual data as personal edited results, and use beam search, sampling or sampling-topk to construct machine translation results. Finally, we used the TER tool [14] to generate word-level `OK/BAD` labels or sentence-level `HTER` scores.

Our system also employs the ensemble strategy to further improve model performance. By training multiple sub-models, the final results are fused by voting or averaging in different tasks.

2 Deep Transformer

A strong and effective feature extraction model is essential for the estimator to make more accurate predictions. We choose the pre-trained machine translation model to extract features. Neural Machine Translation (NMT) based on multi-layer self-attention has shown strong results in many machine translation tasks. In order to improve the performance of machine translation and extract the information contained in the sentences more fully, we apply the structure of Pre-norm Transformer-DLCL. In this section, we describe the details about our deep architecture as below:

Pre-norm Transformer: For Transformer [16], learning deeper networks [1] is not easy because of the difficulty to optimize due to the gradient vanishing/exploring problem. But in recent implementations, Wang et al. [17] emphasized that the location of layer normalization [8] plays a vital role when training deep Transformer. In early versions of Transformer, layer normalization is placed after the element-wise residual addition. While in recent implementations, layer normalization is applied to the input of every sublayer, which can provide a direct way to pass error gradient from top to bottom. In this way pre-norm

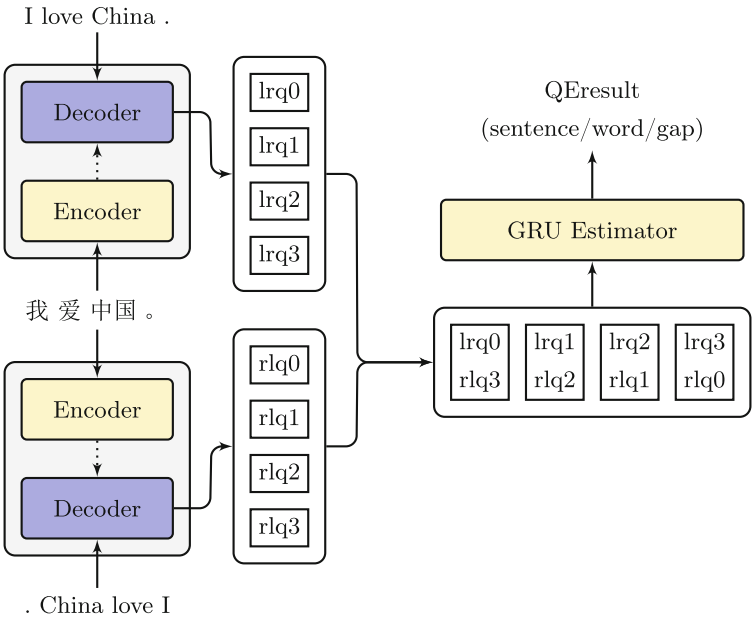

Fig. 1. The architecture of our model based on predictor-estimator.

Transformer is more efficient for training than post-norm (vanilla Transformer) when the model goes deeper.

Transformer-DLCL: In addition, a dynamic linear combination of previous layers method [17] was used in Transformer model. Transformer-DLCL employed direct links with all previous layers and offered efficient access to lower-level representations in a deep stack. An additional weight matrix $W_{l+1} \in R^{L \times L}$ was used to weight each incoming layer in a linear manner. This method can be formulated as:

$$\Psi(y_0, y_1 ... y_l) = \sum_{k=0}^{l} W_k^{l+1} LN(y_k) \tag{1}$$

Equation 1 provided a way to learn preference of layers in different levels of the stack, $\Psi(y_0, y_1 ... y_l)$ was the combination of previous layer representation. Furthermore, this method is model architecture free which can be integrated with either pre-norm Transformer or relative position Transformer [13] for further enhancement. The details can be seen in Wang et al. [17].

We used Transformer-DLCL model with 25 layers in encoder, and show the performance improvement of Transformer-DLCL vs. Transformer-base and Transformer-Big in Table 1.

Table 1. BLEU score and △ BLEU [%] on WMT ZH→EN and EN→ZH *newstest2017*.

Task	Model	BLEU	△BLEU
ZH→EN	Transformer-Base	26.58	–
	Transformer-Big	27.09	+0.51
	Transformer-DLCL-25L	27.55	+0.97
EN→ZH	Transformer-Base	25.54	–
	Transformer-Big	26.59	+1.05
	Transformer-DLCL-25L	27.30	+1.76

3 System

3.1 Architecture

The model architecture of the whole system is presented in Fig. 1. It consists of two parts: a predictor which joint left-to-right and right-to-left Pre-norm Transformer-DLCL, and an estimator with a multi-layer Bi-GRU. Predictor is used to extract semantic information from given machine translation results, according to source-side sentences. In order to fully consider the forward and backward information in the sentences, we use the left-to-right and right-to-left translation models to extract the bidirectional semantic information independently, and then fuse them to obtain the quality vectors. After that, the quality vector is fed into the bidirectional GRU to predict the `HTER` score or `OK/BAD` labels. We first pre-train forward and backward translation models, then jointly train the estimator with the predictor to maximize the evaluation capability of the system.

3.1.1 Deep Bi-Predictor

The sequence-to-sequence based Transformer models [16] are powerful in extracting information and have been proven to be strong in many translation tasks. The Pre-Norm Transformer-DLCL further improves the feature extraction ability. The encoder receives the input sequence $x = \{x_0, x_1...x_n\}$,and maps it to a vector $z = \{z_0, z_1...z_n\}$ of the same length,which contains the source sentence feature. The decoder inputs the translation sequence $y = \{y_0, y_1...y_m\}$ and generates a top-level representation containing sufficient semantic and grammatical information.

Due to the existence of the decoder mask, the unidirectional model can not observe the future information. In order to make the vector extracted by the model contain sufficient context knowledge, we use left-to-right and right-to-left translation models respectively, and extract the feature vectors $l2r$ and $r2l$ independently. We get the final quality vector by concatenating way ($q = [l2r : r2l]$).

3.1.2 Bi-GRU Estimator

RNN is widely used to solve sequence generation problem. And we use a Bi-GRU as our estimator. The Bi-GRU consists of two parts, forward and backward. It reads quality vector q, calculate the forward hidden states $(\overrightarrow{\mathbf{h}}_1, \cdots, \overrightarrow{\mathbf{h}}_T)$ and backward hidden states $(\overleftarrow{\mathbf{h}}_1, \cdots, \overleftarrow{\mathbf{h}}_T)$ respectively, where T is the sequence length. We get the representation of each word by concatenating the forward hidden state $\overrightarrow{\mathbf{h}}_j$ and the backward one $\overleftarrow{\mathbf{h}}_j$, $\boldsymbol{h}_j = [\overrightarrow{\mathbf{h}}_j, \overleftarrow{\mathbf{h}}_j]$. We convert the word-level tasks into classification problems, and Eqs. 2 and 3 show our goals on the word and gap tasks, respectively. Sentence-level tasks are converted to a regression problem, refer to Eq. 4.

$$\arg\min \sum_{j=1}^{T} \textbf{cross\_entropy}\left(y_j, \mathbf{W}_1 \boldsymbol{h}_j\right) \tag{2}$$

$$\arg\min \sum_{j=0}^{T} \textbf{cross\_entropy}\left(y_j, \mathbf{W}_2 \textbf{Conv}(\boldsymbol{h}_j, \boldsymbol{h}_{j+1})\right) \tag{3}$$

$$\arg\min \|h - \textbf{sigmoid}\left(\mathbf{W}_3 \mathbf{h}_T\right)\|_2^2 \tag{4}$$

where h is the real `HTER` score, y_j is real labels, $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{W}_3$ is trainable parameter matrices, and T is the length of the target-side. **cross_entropy** is the cross entropy loss (with logits). **Conv** is a convolution operation that fuses information from adjacent locations for predicting gap tags.

We dynamically control the number of layers of the Bi-GRU according to different data volumes. At the same time, we also try the self-attention layer and self-attention layer + Bi-GRU architectures as estimator, finding there is no better performance. But we use them as candidate models for ensemble to enhance diversity.

3.1.3 BPE Matrix

BPE is introduced to reduce the number of unknown tokens in many NLP tasks. And we also apply it to our model. But there is a problem in word-level task. The length L_b of quality vector extracted by predictor is different from the number L_w of tokens in sentence. We follow Fan et al. [5] to solve this problem by a $L_w \times L_b$ sparse matrix, which average the features of subwords corresponding to one word token, and reduce the length of quality vector from L_b to L_w.

3.2 Data Construction

3.2.1 Bilingual Data for Pre-training

We use WMT 2019 ZH-EN parallel data to pre-train our predictors, which consists of CWMT, wikititles-v1, NewsCommentary-v14 and UN corpus. After filtering, about 11M sentences pair is selected. Furthermore, we use 6M monolingual data from WMT 2019 to construct pseudo data by back-translation [3] in both directions. All parallel data is segmented by NiuTrans [18] word segmentation

toolkit. After the preprocessing, we train BPE [12] models with 32, 000 merge operations for both sides respectively.

3.2.2 Quality Estimation Data

The dataset for QE task consists of three parts: source sentences, machine translations and QEscore (`HTER` score for sentence level or `OK/BAD` labels for word-level). The amount of data provided by CCMT 2019 QE task is no more than 15K. We think it isn't enough to train a strong model, so we construct 50K pseudo data using parallel data from WMT 2019. To obtain high quality bilingual data, we use machine translation model and language model to score parallel data. First, we use the translation model to score the real bilingual data by forced decoding. Secondly, we use the language model to score the source and target sentences, and combine the three scores to sort the real data, select the data with the higher score. After obtaining high-quality bilingual data, we decode them in a variety of ways to obtain machine-translated data, including beam search [11], sampling-topk. We regard the target sentences of bilingual data as personal edited data, and generate the sentence-level `HTER` score or the word-level labels using TER tool [14].

In addition, we find the ratio of `OK/BAD` labels in word gap subtask is about 20:1, which means the `BAD` labels between words corresponding to missing translations is too few and it's hard to predict `BAD` label for trained model. So we randomly drop some word in our machine translation results to improve the number of `BAD` label between words.

3.3 Model Ensemble

In MT systems, ensemble decoding method is wildly used to boost translation quality via integrating the predictions of several single models at each decode step. We try a similar approach in QE task. However, we find that ensemble method is expensive when it comes to more model fusion. It can't try to combine more models in a limited time, so we adopt an external fusion method:

- We select twelve high-scoring single models using different model architectures or datasets, and decode 12 results as candidates.
- Calculate all combinations of twelve models externally.
- For different combinations, word-level tasks, we use the voting method to ensemble, and the sentence-level we average `HTER` score.
- Pick the best performing model combination.

In this way, we quickly try out all the combinations of candidates in a short time, and it is easier to pick the optimal combination.

4 Experiments and Results

We implement our QE models based on Fairseq [10]. Transformer-DLCL models are pre-trained on eight 1080Ti GPUs. We use the Adam optimizer with $\beta_1 = 0.97$, $\beta_2 = 0.997$ and $\epsilon = 10^{-6}$. The training data is reshuffled after finishing

Table 2. Word-level word result on CCMT QE *valid2019*. We use a jointly l2r and r2l Transformer-DLCL as a predictor and Bi-GRU as an estimator to jointly train with different datasets.

Construction method	F1-OK	F1-BAD	F1-multi
–	0.8353	0.5673	0.4739
High quality bilingual	0.8642	0.5897	0.5096
Bilingual-beam	0.8735	0.5747	0.502
Bilingual-sampling-topk	0.8691	0.5795	0.5036
Bilingual-round-trip	0.8632	0.5833	0.5035

each training epoch, and we batch sentence pairs by target-side sentences lengths, with 8192 tokens per GPU. Large learning rate and warmup-steps are chosen for faster convergence. We set max learning rate as 0.002 and warmup-steps as 8000. For the jointly training predictor-estimator architecture, we train it on one 1080Ti GPU, 1024 tokens per step. And we set max learning rate as 0.0005 and warmup-steps as 200.

Moreover, due to the lack of `BAD` labels in the word-level tasks are relatively small, the model tends to predict all labels as `OK` in the inference stage. So we introduce the bad-enhanced parameter, strengthen the weight of the `BAD` label when calculating the loss, thereby improving the ability of the model to predict `BAD`. Next, we will show details in the following subsections.

4.1 QE Pseudo Data

We compare different method on the task of ZH2EN word-level. The following will introduce the method we use.

- Use high-quality bilingual data such as newtest2016, newtest2017, and use the target as the result of personal editing, and decode the source to construct dataset by sampling-topk.
- The data selected from the bilingual data, and the pseudo datasets decoded by the beam search [11] or the Sampling-topk.
- We translate the monolingual data in target side to the source sentences, and then translate generated sentences back to target side, this method names round-trip [6]. The detail results are shown in Table 2.

The round-trip and sampling-topk methods are mainly aimed at the unbalanced distribution of `OK` and `BAD` labels in word-level tasks. We increase the number of BAD tags by introducing noise during the decoding process. The Table 2 shows that pseudo-data using high-quality bilingual constructs delivers the greatest performance improvement in the same architecture. However, there are no significant differences in the average label distribution in the results by introducing noise in a variety of ways. We speculate that the target language in

Table 3. Word-level result on CCMT QE *valid2019*. We use GRU as an estimator to jointly train using officially available data.

Task	Precitor	F1-OK	F1-BAD	F1-multi
ZH2EN word-level word	Transformer-base	0.8932	0.4618	0.4125
	Transformer-Big	0.8946	0.4709	0.4212
	Deep Transformer-DLCL	0.8477	0.5078	0.4305
ZH2EN word-level gap	Transformer-base	0.9511	0.1682	0.1600
	Transformer-Big	0.9516	0.1976	0.1881
	Deep Transformer-DLCL	0.9552	0.1981	0.1892
EN2ZH word-level word	Transformer-base	0.8896	0.4043	0.3597
	Transformer-Big	0.8904	0.4176	0.3718
	Deep Transformer-DLCL	0.8727	0.4309	0.3761
EN2ZH word-level gap	Transformer-base	0.9585	0.1454	0.1394
	Transformer-Big	0.9472	0.149	0.1411
	Deep Transformer-DLCL	0.9493	0.1533	0.1455

high-quality bilingual data is closer to the personal editing results, and the generated tags are more consistent with the real data, which makes the model more accurate. Different datasets are also used to increase data diversity in model fusion.

4.2 Different Predictor

Our model base on the predictor-estimator architecture. Recent research shows that the Transformer [16] has powerful information extraction capability. Therefore, we use the translation model as a predictor to extract the semantic information contained in the sentence. At the same time, we empirically believe that a stronger translation model can bring greater performance improvement to the QE task. In order to verify the impact of the pre-trained translation model on the QE model, we conduct multiple experiments with different left-to-right predictors and the same estimator. The result of word-level is shown on Table 3, Sentence-level on Table 4.

From the Tables 3 and 4, we find the estimator has better performance with more powerful translation model.

4.3 Different Estimator

After determining the architecture of the predictor, we try a variety of architectures as the estimator, including GRU, Bi-GRU and self-attention. We take the task of the ZH-EN word-level as an example. In Table 5, we show different prediction results in different architectures.

Table 4. Sentence-level result on CCMT QE *valid2019*. We use GRU as an estimator to jointly train using officially available data.

Task	Precitor	Person's
ZH2EN sentence-level	Transformer-base	0.5548
	Transformer-Big	0.5645
	Transformer-DLCL	0.5699
EN2ZH sentence-level	Transformer-base	0.4696
	Transformer-Big	0.4872
	Transformer-DLCL	0.5071

Table 5. ZH2EN word-level word result on CCMT QE *valid2019*. We use a jointly l2r and r2l Transformer-DLCL as a predictor.

Estimator	F1-OK	F1-BAD	F1-multi
GRU	0.8731	0.5427	0.4738
Bi-GRU	0.8642	0.5897	0.5096
Self-attention	0.8165	0.5265	0.4299
Self-attention + Bi-GRU	0.8511	0.5519	0.4697

We use real data and high-quality bilingual constructed pseudo-data total 30k as jointly training data. We can observe that Bi-GRU performs significantly better than other architectures with the same dataset. However, due to the possibility of data scarcity that makes complex architecture trained inadequately, we also try to increase the amount of pseudo-data for the self-attention layer and self-attention + Bi-GRU architecture. We found that increasing the amount of data lead to the performance improvement of more complex estimator architectures. But it's still a little worse than the Bi-GRU. We use them as seed models for system integration to increase diversity.

4.4 Ensemble

We construct multiple sub-models through different model architectures and data sets, and integrate the results of multiple systems externally on all tasks to further improve the stability and performance of the system. We use the left-to-right Transformer-DLCL as the predictor and the GRU as the estimator to build our baseline system. Table 6 shows the final results of all of our participating tasks.

Table 6. All word-level and sentence-level result on CCMT QE *valid2019*.

System	ZH2EN						
	Word-level word			Word-level gap			Sentence-level
	F1-OK	F1-BAD	F1-multi	F1-OK	F1-BAD	F1-multi	Person's
Baseline	0.8477	0.5078	0.4305	0.9552	0.1981	0.1892	0.5699
+Bi-GRU	0.8673	0.5215	0.4523	0.9556	0.2116	0.2022	0.5802
+r2l predictor	0.8353	0.5673	0.4739	0.9570	0.2583	0.2472	0.5831
+Pseudo data	0.8642	0.5897	0.5096	0.9615	0.2776	0.2669	0.5830
+Ensemble	0.8767	0.6152	0.5393	0.9622	0.2887	0.2778	0.6164
System	EN2ZH						
	Word-level word			Word-level gap			Sentence-level
	F1-OK	F1-BAD	F1-multi	F1-OK	F1-BAD	F1-multi	Person's
Baseline	0.8727	0.4309	0.3761	0.9493	0.1533	0.1455	0.5071
+Bi-GRU	0.8932	0.4692	0.4199	0.9671	0.1669	0.1614	0.5501
+r2l predictor	0.898	0.4695	0.4217	0.9596	0.179	0.1718	0.5537
+Pseudo data	0.8941	0.4762	0.4258	0.9656	0.2083	0.2011	0.5491
+Ensemble	0.8974	0.4886	0.4385	0.9715	0.2283	0.2218	0.5861

5 Conclusion

This paper describes our systems for CCMT19 Quality Estimate tasks including both word-level and sentence-level.

We adopt predictor-estimator architecture, use Transformer-DLCL as Predictor based on deep network [1], and combine left-to-right and right-to-left models to further enhance predictor's feature extraction capabilities. Estimator adopts the Bi-GRU and uses the quality vector extracted by predictor to predict for different tasks.

At the same time, we further improve the performance of the translation model as predictor and the prediction performance of estimator by artificially constructing pseudo-data. In addition, a external ensemble algorithm is helpful to search a robust combination of models.

Acknowledgments. This work was supported in part by the National Science Foundation of China (Nos. 61876035, 61732005 and 61432013), the National Key R&D Program of China (No. 2019QY1801) and the Opening Project of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research. We also thank the reviewers for their insightful comments.

References

1. Bapna, A., Chen, M.X., Firat, O., Cao, Y., Wu, Y.: Training deeper neural machine translation models with transparent attention. arXiv preprint arXiv:1808.07561 (2018)
2. Blatz, J., et al.: Confidence estimation for machine translation. In: Coling 2004: Proceedings of the 20th International Conference on Computational Linguistics (2004)
3. Douglas, S.P., Craig, C.S.: Collaborative and iterative translation: an alternative approach to back translation. J. Int. Mark. **15**(1), 30–43 (2007)
4. Edunov, S., Ott, M., Auli, M., Grangier, D.: Understanding back-translation at scale. arXiv preprint arXiv:1808.09381 (2018)
5. Fan, K., Li, B., Zhou, F., Wang, J.: "Bilingual expert" can find translation errors, July 2018
6. Junczys-Dowmunt, M., Grundkiewicz, R.: Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In: WMT (2016)
7. Kim, H., Jung, H.Y., Kwon, H., Lee, J.H., Na, S.H.: Predictor-estimator: neural quality estimation based on target word prediction for machine translation. ACM Trans. Asian Low-Resour. Lang. Inf. Process. (TALLIP) **17**(1), 3 (2017)
8. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
9. Niehues, J., Herrmann, T., Vogel, S., Waibel, A.: Wider context by using bilingual language models in machine translation. In: Proceedings of the Sixth Workshop on Statistical Machine Translation, pp. 198–206. Association for Computational Linguistics (2011)
10. Ott, M., et al.: fairseq: A fast, extensible toolkit for sequence modeling. In: Proceedings of NAACL-HLT 2019: Demonstrations (2019)
11. Sennrich, R., Haddow, B., Birch, A.: Improving neural machine translation models with monolingual data. arXiv preprint arXiv:1511.06709 (2015)
12. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909 (2015)
13. Shaw, P., Uszkoreit, J., Vaswani, A.: Self-attention with relative position representations. arXiv preprint arXiv:1803.02155 (2018)
14. Snover, M., Dorr, B., Schwartz, R., Micciulla, L., Makhoul, J.: A study of translation edit rate with targeted human annotation. In: Proceedings of Association for Machine Translation in the Americas, vol. 200 (2006)
15. Specia, L., Paetzold, G., Scarton, C.: Multi-level translation quality prediction with quest++. In: Proceedings of ACL-IJCNLP 2015 System Demonstrations, pp. 115–120 (2015)
16. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 6000–6010 (2017)
17. Wang, Q., et al.: Learning deep transformer models for machine translation. arXiv preprint arXiv:1906.01787 (2019)
18. Xiao, T., Zhu, J., Zhang, H., Li, Q.: Niutrans: an open source toolkit for phrase-based and syntax-based machine translation. In: Proceedings of the ACL 2012 System Demonstrations ACL 2012, pp. 19–24. Association for Computational Linguistics, Stroudsburg, PA, USA (2012). http://dl.acm.org/citation.cfm?id=2390470.2390474