# Efficient Transfer Learning for Quality Estimation with Bottleneck Adapter Layer

**Hao Yang, Minghan Wang, Ning Xie, Ying Qin, Yao Deng**
Huawei Translation Service Center, Beijing, China
`{yanghao30,wangminghan,nicolas.xie,qinying,dengyao3}@huawei.com`

## Abstract

The Predictor-Estimator framework for quality estimation (QE) is commonly used for its strong performance, where the predictor and estimator works on feature extraction and quality evaluation, respectively. However, training the predictor from scratch is computationally expensive. In this paper, we propose an efficient transfer learning framework to transfer knowledge from NMT dataset into QE models. A Predictor-Estimator alike model named BAL-QE is also proposed, aiming to extract high quality features with pre-trained NMT model, and make classification with a fine-tuned Bottleneck Adapter Layer (BAL). The experiment shows that BAL-QE achieves 97% of the SOTA performance in WMT19 En-De and En-Ru QE tasks by only training 3% of parameters within 4 hours on 4 Titan XP GPUs. Compared with the commonly used NuQE baseline, BAL-QE achieves 47% (En-Ru) and 75% (En-De) of performance promotions.

## 1 Introduction & Related work

Translation quality estimation (QE) has become one of the important research topics in the discipline of machine translation (MT). QE aims to solve the problem of how to evaluate the quality of the translation results and predict the types of errors and locations (Specia et al., 2013), with only source sentences and machine translation results, without the post edited reference. (Junczys-Dowmunt, 2019; Yang et al., 2019b; Yang et al., 2019a). QE tasks can be divided into word level, phrase level and sentence level. In this paper, we only focus on word-level QE tasks.

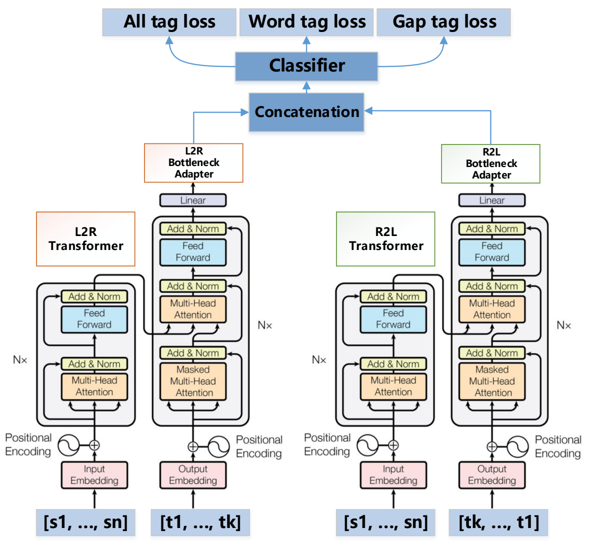

**Figure 1:** The architecture of BAL-QE, where two Transformers are used to produce features in both direction, then, being processed by dual Bottleneck Adapters and fed into classifiers.

There are two main categories of neural network machine translation quality estimation systems, end-to-end neural network framework and two-stage neural network architecture. A representative architecture of the first one is named Neural QE (NuQE) (Kreutzer et al., 2015; Martins et al., 2016), which directly predicts sequence labels by passing source and MT results into a unified model composed with several bi-LSTM layers. The other one is Predictor-Estimator architecture (Kim and Lee, 2016; Kim et al., 2017; Wang et al., 2018; Li et al., 2018), which is composed of two subsequent neural models: 1) a word prediction model that

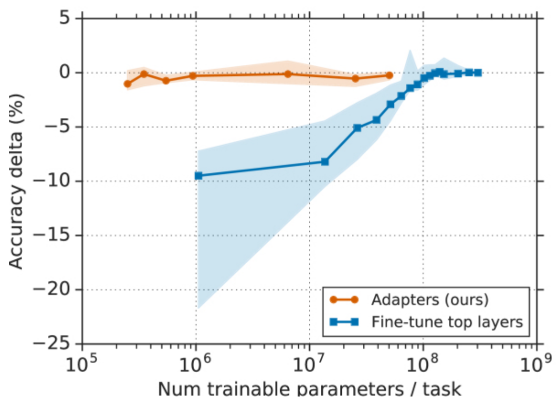

**Figure 2:** (Houlsby et al., 2019) The performance gain of the transfer learning on 18 GLUE corpus, which is based on adapters, which achieves 99.6% of SOTA performance by only adding 3% of training parameters

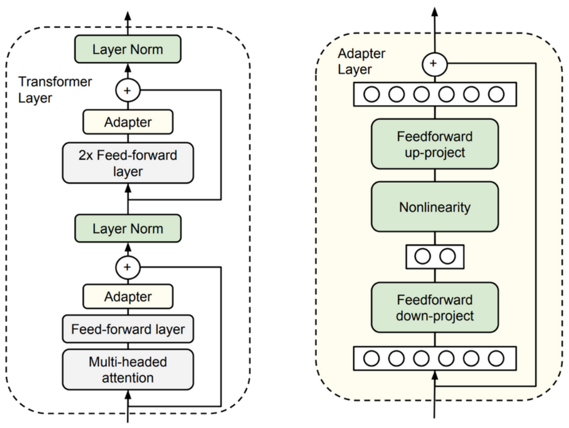

**Figure 3:** (Houlsby et al., 2019) Bottleneck Adapter Layer for Transformer fine-tuning. Only three green parts in the right, including Feedforward down-project, Nonlinearity and Feedforward up-project, are need to be trained, the parameters of the left transformer are fixed, total training parameters ratio is 3%.

predicts each word given the left and right context of the source and target corpus, and 2) a quality estimation model, which estimates word-level labels based on the features generated by the predictor. Because the predictor itself can be regarded as a neural machine translation (NMT) system, which can be trained based on a large volume of external parallel corpora and provides high quality semantic features, Predictor-Estimator framework is much better than NuQE.

Transfer learning (TL) or fine-tuning large pre-trained language models (PLMs) is an effective method in NLP, which can produce strong performance on many NLP tasks (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018). There are two types of transfer learning. The first one is full-parameter fine-tuning with in-domain data, which aims to fit the distribution of in-domain data without damaging out-domain performance. The other one is to add additional layers to the original architecture as adapters and only update those newly added layers, resulting in a significant speed-up for fine-tuning. The Bottleneck Adapter Layer (BAL) (Houlsby et al., 2019; Rebuffi et al., 2017) proposed by Google in 2019, shows that BAL-based transfer learning could obtain 99.6% of the SOTA performance by only training 3% of the parameters.

The contribution of our paper is as follows:

- We propose an efficient transfer learning framework which transfers knowledge learned from NMT tasks to QE tasks by fine-tuning the pre-trained NMT model with QE data.
- We propose the BAL-QE which achieves 97% of the SOTA performance by only training the Bottleneck layer which is equivalent to 3% of parameters of the entire model, and converges within 4 hours. The model is open-sourced.

## 2 Modelling of BAL-QE

### 2.1 Modelling of QE

For a word-level QE task, tokens correctly translated should be tagged as OK, while mistranslated or ignored tagged as BAD. Besides, there should be tags for gaps. We consider gaps as the position between each two words. Words correctly aligned with the source are tagged as OK, otherwise as BAD. If one or more words are missing in the translation, their positions (gap) are tagged as BAD, and OK otherwise. (Wang et al., 2019).

More formally, QE can be considered as taking two sequences as inputs (i.e. source text and the translated text (MT) required for evaluation) and outputs a single sequence (i.e. tags), as shown in Figure 4. When there are $K$ tokens in MT, the word tag should have same length, and the gap tag should have a length of $K+1$ which is the number of positions between two words as well as the beginning and the end. The length of all tags is $2K+1$, representing the combination of word and gap tags. Here, we define the QE system as a function $f$:

$$[e_1, ..., e_{2K+1}] = f([x_1, ..., x_m], [\hat{y}_1, ..., \hat{y}_k]) \quad (1)$$

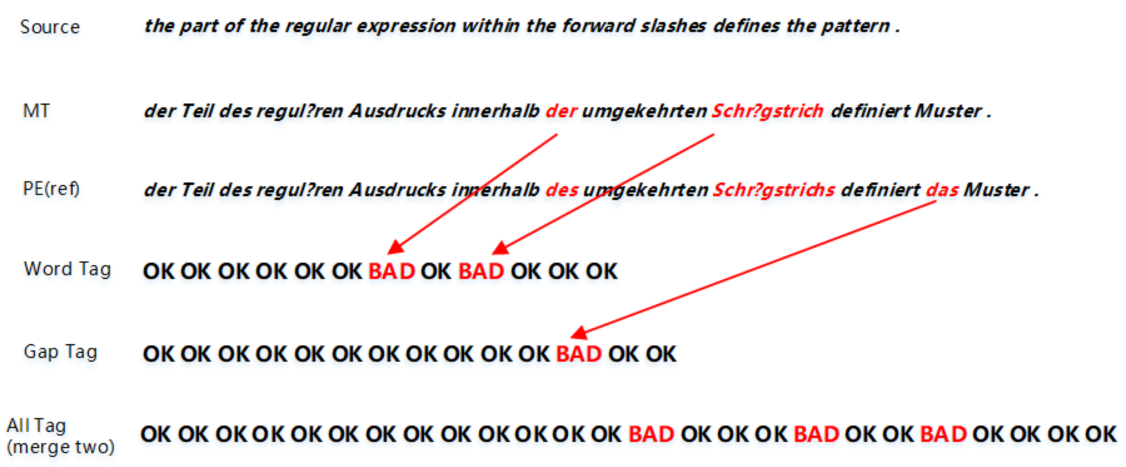
Source
the part of the regular expression within the forward slashes defines the pattern .
MT
der Teil des regul?ren Ausdrucks innerhalb der umgekehrten Schr?gstrich definiert Muster .
PE(ref)
der Teil des regul?ren Ausdrucks innerhalb des umgekehrten Schr?gstrichs definiert das Muster .
Word Tag
OK OK OK OK OK OK BAD OK BAD OK OK OK
Gap Tag
OK OK OK OK OK OK OK OK OK OK OK BAD OK OK
All Tag (merge two)
OK OK OK OK OK OK OK OK OK OK OK OK OK BAD OK OK OK BAD OK OK BAD OK OK OK OK

**Figure 4:** An example of QE, where the word tag represents for whether the predicted token is correct, the gap tag means whether there are missing words between two predicted words. All tag is the staggered arrangements of the word and the gap tag.

where $e$ represents tags, $x$ is source text and $\hat{y}$ is the translation. We stagger the word tags and gap tags one by one to create the all tag sequence, where even indices are word tags and odd indices are gap tags (counting from 1). For a word tag, if the tag is BAD, it means the translated word is incorrect or has to be deleted. For a gap tag, if the tag is BAD, it means there are missing words in the gap.

## 2.2 Optimized Loss Function

With the improvement of the performance of NMT systems, the proportion of BAD tags becomes much fewer than OK tags in QE corpus. Therefore the loss function has to be optimized to handle such imbalance. We optimize the imbalance from three aspects: 1) Improving the effect of BAD tags on the model. 2) Optimizing three losses with appropriate weights. 3) Applying MCC as evaluation metrics to obtain reasonable results.

To improve the effect of BAD tags, we use a hyper-parameter $\alpha$ in the loss function to control the punishments of incorrect prediction of BAD tags. The newly introduced loss is denoted as follows:

$$\mathcal{L}^* = \begin{cases} -[y \log p + (1-y)\log(1-p)], & \text{if } y = 1 \\ -\alpha[y \log p + (1-y)\log(1-p)], & \text{if } y = 0 \end{cases} \tag{2}$$

where $y = 1$ represents for OK tag and $y = 0$ represents for BAD tag. The $\alpha$ is set as 9 in the experiment due to the ratio of OK and BAD is 0.88:0.12 and 0.93:0.07 for word and gap tag respectively (Wang et al., 2018; Wang et al., 2019).

Apart from the imbalance optimized loss, we also use multi-task learning to optimize the model by simultaneously optimizing the loss of words, gaps and all tags. The merged loss is represented as:

$$\mathcal{L} = \sum_{t \in \mathcal{T}} \lambda_t \mathcal{L}_t^* \tag{3}$$

where $\mathcal{T} = \{\text{all-tag}, \text{word-tag}, \text{gap-tag}\}$, and $\sum_{t \in \mathcal{T}} \lambda_t = 1$.

## 2.3 Evaluation Metrics

QE can be considered as a sequential labelling problem with two classes. A fine-grained F1-score and MCC are used to evaluate the results because of the imbalance. The fine-grained F1-score is composed of $\text{F1}_{\text{all}}$, $\text{F1}_{\text{word}}$ and $\text{F1}_{\text{gap}}$. For each F1, it can be computed as $\text{F1}_{\text{t}} = \text{F1}_{\text{t-OK}} \times \text{F1}_{\text{t-BAD}}$, $t \in \mathcal{T}$. The F1 is calculated as standard form: F1=(2 $\times$ precision $\times$ recall) / (precision + recall)

Additionally, we use Matthews correlation coefficient (MCC) for producing unbiased evaluations over the unbalanced predictions. MCC is computed as follows:

$$S = \frac{TP + FN}{N} \tag{4}$$

$$P = \frac{TP + FP}{N} \tag{5}$$

$$MCC = \frac{\frac{TP}{N} - SP}{\sqrt{SP(1-S)(1-P)}} \tag{6}$$

## 2.4 Model Architecture of BAL-QE

When applying transfer learning on QE, we need a pre-trained NMT model and an adapter layer for

|  |  | MT(ALL) |  |  |  | MT(Word) |  |  |  | MT(Gap) |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | $F1_{all}$ | F1-BAD | F1-OK | MCC | $F1_{word}$ | F1-BAD | F1-OK | MCC | $F1_{gap}$ | F1-BAD | F1-OK | MCC |
| EN_RU | UNBABEL | **0.45961** | **0.478018** | **0.960251** | **0.401625** | **0.48894** | **0.529076** | **0.924197** | **0.418838** | **0.18664** | **0.189196** | **0.984127** | **0.1836** |
|  | ETRI | 0.3895 | 0.4051 | 0.9617 | 0.3325 | 0.4215 | 0.4561 | 0.924 | 0.34675 | 0.1609 | 0.1631 | 0.9803 | 0.152 |
|  | baseline | 0.2412 | 0.250005 | 0.9325 | 0.2145 | 0.222286 | 0.284211 | 0.914 | 0.21913 | 0.101053 | 0.102 | 0.932 | 0.096 |
|  | Uni BAL-QE | 0.35055 | 0.36459 | 0.942 | 0.29925 | 0.37935 | 0.41049 | 0.922 | 0.312075 | 0.1358 | 0.14679 | 0.972 | 0.1368 |
|  | Bi BAL-QE | **0.424555** | **0.441559** | **0.96098** | **0.367063** | **0.45522** | **0.492588** | **0.924098** | **0.382794** | **0.1876** | **0.176148** | **0.982213** | **0.1678** |
| EN_DE | UNBABEL | **0.4523324** | **0.47** | **0.962** | **0.380471** | **0.495305** | **0.5336** | **0.933962** | **0.367166** | **0.313835** | **0.317975** | **0.987382** | **0.2737** |
|  | ETRI | 0.4028 | 0.4198 | 0.9595 | 0.342088 | 0.4307 | 0.464 | 0.9283 | 0.319275 | 0.2729 | 0.2765 | 0.9871 | 0.238 |
|  | baseline | 0.2974 | 0.311702 | 0.954984 | 0.254 | 0.319795 | 0.34452 | 0.927326 | 0.237062 | 0.202628 | 0.205301 | 0.985366 | 0.175172 |
|  | Uni BAL-QE | 0.346408 | 0.361028 | 0.955679 | 0.294195 | 0.370402 | 0.39904 | 0.948718 | 0.274577 | 0.234694 | 0.23779 | 0.987342 | 0.20468 |
|  | Bi BAL-QE | **0.4275662** | **0.4449** | **0.96075** | **0.361279** | **0.463003** | **0.4988** | **0.931131** | **0.343221** | **0.293368** | **0.297238** | **0.987241** | **0.25585** |

**Table 1:** The experimental result, note that top-2 results are bold. $F1_{all}$, $F1_{word}$, $F1_{tag}$ are the multiplication of F1-OK and F1-BAD in specific level.

| Split | Pair | Sentences | Words | BAD source | BAD target | HTER |
|---|---|---|---|---|---|---|
| Train | EN-DE | 13,442 | 234,725 | 28,549(12.16%) | 37.040(7.06%) | 0.15(±0.19) |
|  | EN-RU | 15,089 | 148,551 | 15,599 (10.50%) | 18,380 (6.15%) | 0.13 (±0.24) |
| Dev | EN-DE | 1,000 | 17,669 | 2,113 (11.96%) | 2,654 (6.73%) | 0.15 (±0.19) |
|  | EN-RU | 1,000 | 9,710 | 1,055 (10.87%) | 1,209 (6.17%) | 0.13 (±0.23) |
| Test | EN-DE | 1,023 | 17,649 | 2,415 (13.68%) | 3,136 (8.04%) | 0.17 (±0.19) |
|  | EN-RU | 1,023 | 7,778 | 1,049 (13.49%) | 1,165 (7.46%) | 0.17 (±0.28) |

**Table 2:** The detail of WMT19 QE dataset

downstream tasks. However, different from original MT tasks which generate tokens depending on previous history, the input of QE is a known sequence which means that when evaluating the token in the current step, we can use future contexts. Therefore, we propose the BAL-QE model which contains three parts: 1) Two pre-trained NMT models, $M_{\mathrm{L2R}}$ and $M_{\mathrm{R2L}}$. 2) Two Bottleneck Adapters for decoders of $M_{\mathrm{L2R}}$ and $M_{\mathrm{R2L}}$. 3) A classifier layer.

The two pre-trained NMT models are Transformer-big (Ng et al., 2019; Junczys-Dowmunt, 2019), including 6 encoders and 6 decoders composed of multi-head self-attentions and cross-attentions. The only difference of the two Transformers used in BAL-QE is the generating direction.

As shown in Figure 3, the Bottleneck Adapter is like an auto-encoder (Houlsby et al., 2019; Artetxe and Schwenk, 2019; Howard and Ruder, 2018; Rebuffi et al., 2017), which is composed of three parts: 1) The feed-forward down-project, which maps the input vector into low-dimensional space. 2) The nonlinear layer, which is actually is an activation function. 3) The feed-forward up-project, which recovers the vector back to high-dimensional space. 4) A residual connection between the inputs and outputs.

The last classifier layer is a linear layer, which takes the concatenated output vectors from two adapters as input, and makes binary classification of each tag. Not surprisingly, we find that bidirectional predictor (dual Transformer) could improve 8% of the performance compared with unidirectional predictor (single Transformer).

## 3 Experiment

### 3.1 Dataset

The Dataset used in the experiment is from WMT19 Quality Evaluation Task1, including two languages (En-De, En-Ru). There are 13,000 sentence pairs for En-De, with approximately 234,000 tokens. The proportion of BAD tag in German MT sentences is 7.06%. En-Ru contains totally 15,000 sentence pairs with 148,000 tokens and 6.15% of BAD tags. More details are shown in Table 2.

### 3.2 Setup of Pre-training Two Transformers

The pre-training of the Transformer is similar with the setup of FAIR SOTA model in WMT19 (Ng et al., 2019), which is implemented with fairseq[1]. BPE is used for tokenizing, where 32000 tokens are reserved. We use UN corpus and Common Crawl parallel corpus with the size of

[1]https://github.com/pytorch/fairseq

|  | Total Params | Training Params | Training Ratio |
|---|---|---|---|
| Uni BAL-QE | 216,235,012 | 6,323,204 | 2.92% |
| Bi BAL-QE | 432,470,002 | 12,646,406 | 2.92% |

**Table 3:** The comparison of parameters of BAL-QE

27,000,000. We also use back-translation to produce 20,000,000 augmented corpus. The BLEU of $M_{\mathrm{L2R}}$ and $M_{\mathrm{R2L}}$ are 42.3 and 41.8 for EN-DE, 36.2 and 35.9 for EN-RU respectively, with less than 2% of difference compared with the SOTA result of published fairseq implementation.

### 3.3 Setup of Fine-tuning BAL-QE

In the fine-tuning of BAL-QE, the parameter of two Transformers are fixed, and we only update the two adapters as well as the classifier, which means that only 2.92% of parameters are trained in the fine-tuning, as shown in Table 3. Adam is used as the optimizer with a triangular learning rate schedule with peak learning rate as 5e-5. We use a maximum of 1,024 tokens per batch and save checkpoints every 1,000 steps, on the exponential moving averaged parameters (Junczys-Dowmunt, 2019) with a decay rate of 1e-4. BPE is applied with subword-nmt, and 32,000 tokens are reserved. It takes 2 hours and 38 minutes and 4 hours and 02 minutes to train the unidirectional and bidirectional BAL-QE on 4 Titan XP GPUs, respectively.

### 3.4 Analysis

As shown in Table 1, MT (ALL), MT (Word) and MT (Gap) represents evaluation results of All Tag, Word Tag and Gap Tag, respectively. The baseline is a model of NuQE. On En-De and Ee-Ru datasets, the unidirectional BAL-QE improves performance by 17% and 45%, and the bidirectional BAL-QE improves by 44% and 75%, compared with the baseline. All metrics of bidirectional BAL-QE achieves top-2 rank, and the F1-OK of En-Ru achieves the SOTA result.

## 4 Conclusion

This paper proposes a Predictor-Estimator QE model based on the Bottleneck Adapter Layer and the Transformer. An efficient transfer learning framework is also proposed, which could transfer knowledge learned from NMT parallel corpora into the QE task to improve the training efficiency of the proposed BAL-QE model. Experiments shows that partially training the model (estimator) could effectively speed up the training and achieves 97% of the SOTA performance.

## References

Artetxe, Mikel and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Dai, Andrew M and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. *arXiv preprint arXiv:1902.00751*.

Howard, Jeremy and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Junczys-Dowmunt, Marcin. 2019. Microsoft translator at wmt 2019: Towards large-scale document-level neural machine translation. *arXiv preprint arXiv:1907.06170*.

Kim, Hyun and Jong-Hyeok Lee. 2016. Recurrent neural network based translation quality estimation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 787–792.

Kim, Hyun, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 562–568.

Kreutzer, Julia, Shigehiko Schamoni, and Stefan Riezler. 2015. Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 316–322.

Li, Maoxi, Qingyu Xiang, Zhiming Chen, and Mingwen Wang. 2018. A unified neural network for quality estimation of machine translation. *IEICE TRANSACTIONS on Information and Systems*, 101(9):2417–2421.

Martins, André FT, Ramón Astudillo, Chris Hokamp, and Fabio Kepler. 2016. Unbabel's participation in the WMT16 word-level translation quality estimation shared task. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 806–811.

Ng, Nathan, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 News Translation Task Submission. *arXiv preprint arXiv:1907.06616*.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Rebuffi, Sylvestre-Alvise, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516.

Specia, Lucia, Kashif Shah, José GC De Souza, and Trevor Cohn. 2013. QuEst-A translation quality estimation framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84.

Wang, Jiayi, Kai Fan, Bo Li, Fengming Zhou, Boxing Chen, Yangbin Shi, and Luo Si. 2018. Alibaba submission for WMT18 quality estimation task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 809–815.

Wang, Ziyang, Hui Liu, Hexuan Chen, Kai Feng, Zeyang Wang, Bei Li, Chen Xu, Tong Xiao, and Jingbo Zhu. 2019. NiuTrans Submission for CCMT19 Quality Estimation Task. In *China Conference on Machine Translation*, pages 82–92. Springer.

Yang, Hao, Gengui Xie, Ying Qin, and Song Peng. 2019a. Domain Specific NMT based on Knowledge Graph Embedding and Attention. In *2019 21st International Conference on Advanced Communication Technology (ICACT)*, pages 516–521. IEEE.

Yang, Muyun, Xixin Hu, Hao Xiong, Jiayi Wang, Yiliyaer Jiaermuhamaiti, Zhongjun He, Weihua Luo, and Shujian Huang. 2019b. CCMT 2019 Machine Translation Evaluation Report. In *China Conference on Machine Translation*, pages 105–128. Springer.