DirectQE: Direct Pretraining for Machine Translation Quality Estimation

Qu Cui¹, Shujian Huang¹, Jiahuan Li¹, Xiang Geng¹, Zaixiang Zheng¹, Guoping Huang², Jiajun Chen¹
¹National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
²Tencent AI Lab, Shenzhen, China

cuiq, lijh, gx, zhengzx}@smaill.nju.edu.cn, {huangsj, chenjj}@nju.edu.cn, donkeyhuang@tencent.com

Abstract

Machine Translation Quality Estimation (QE) is a task of predicting the quality of machine translations without relying on any reference. Recently, the predictor-estimator framework trains the predictor as a feature extractor, which leverages the extra parallel corpora without QE labels, achieving promising QE performance. However, we argue that there are gaps between the predictor and the estimator in both data quality and training objectives, which preclude QE models from benefiting from a large number of parallel corpora more directly. We propose a novel framework called DirectQE that provides a direct pretraining for QE tasks. In DirectQE, a generator is trained to produce pseudo data that is closer to the real QE data, and a detector is pretrained on these data with novel objectives that are akin to the QE task. Experiments on widely used benchmarks show that DirectQE outperforms existing methods, without using any pretraining models such as BERT. We also give extensive analyses showing how fixing the two gaps contributes to our improvements.

Introduction

Evaluating the results of Machine Translation (MT) usually requires human translations as references, which are expensive to obtain (Papineni et al. 2002; Denkowski and Lavie 2014). Quality Estimation (QE) aims to directly predict the quality of translations without relying on any reference. An example is shown in Table 1.

| Source | Translation | HTER |
|--------|-------------|------|
| anschließend wird in jeder Methode die übergeordnete Superclass-Version von selbst aufgerufen. | anschließend wird in jeder Methode die übergeordnete Superclass-Version von selbst aufgerufen. | 0.2727 |

Table 1: An example of QE. Word-level QE assigns labels to each token/word. For example, some tokens are tagged as ‘BAD’ (marked underline), and the rest are tagged as ‘OK’. Sentence-level QE labels the whole sentence with a single quality score, representing the effort to correct the translation manually.

After those methods relying on hand-crafted features (Kozlova, Shmatova, and Frolov 2016; Sagemo and Stymne 2016), neural QE approaches show promising achievements (Chen et al. 2017; Shah et al. 2016; Abdelsalam, Bojar, and El-Beltagy 2016). Because labeled QE data is hard to obtain and often limited in size, it is natural to transfer bilingual knowledge from parallel data of the MT tasks (Kepler et al. 2019). One well-known framework for this knowledge transfer is the predictor-estimator framework (Kim, Lee, and Na 2017).

In the predictor-estimator framework, a predictor is trained on parallel data to extract bilingual features. The training objective is to predict one target token given the source sentence and the rest tokens in the reference (Kepler et al. 2019; Kim et al. 2019). Then, an estimator will be trained on the amount-limited real QE data to make quality estimations, based on features (Fan et al. 2018) provided by the predictor.

However, we suggest that the ability of the estimator is limited due to the limited amount of labeled data. On the other hand, there are two major gaps between the predictor and the estimator, which may hinder the transfer of bilingual knowledge. Firstly, the predictor is trained on the well-translated parallel data for MT. In contrast, the estimator aims to work for the QE task, which deals with imperfect translations from MT systems as the input. The differences in data quality may lead to a degradation of QE performance (Shimodaira 2000; Weiss, Khoshgoftaar, and Wang 2016).

Secondly, the predictor is trained with the objective to make token predictions, but the estimator is trained to do QE tasks, which discriminate different translation qualities. The features used to predict tokens may not be suitable to judge the translation qualities. Moreover, during the training of the predictor on parallel data, there is no direct supervision from QE tasks. Hence, it is hard for the predictor to learn QE-related features that the estimator can directly exploit.

In this paper, we propose a novel framework called DirectQE that provides a direct pretraining for QE tasks. In DirectQE, a proposed generator is first trained on parallel data and then be used to produce a large amount of pseudo

¹The predictor could also be finetuned on the QE data.
QE data. Our design of the generator makes the generated translations more similar to the real QE data than the parallel data itself and makes it possible to generate the QE labels automatically. Then a proposed detector learns quality estimation by pretraining on the pseudo QE data and finetuning on real QE data. We design specific pretraining objectives so that the training on the pseudo and real QE data could be well connected.

Unlike the previous estimator, which could only be trained on limited QE data, the detector in our framework enables a direct usage of a large amount of pseudo QE data. Besides, instead of relying on features from a QE-irrelevant predictor, our detector performs the quality predictions using features, learned through the direct pretraining, in its own way. We conduct extensive experiments to show how fixing these two gaps can contribute to QE tasks. With the same amount of the extra parallel dataset, DirectQE achieves new SOTA results on different QE datasets.

**Background**

**Formalization of QE Task**

Machine Translation Quality Estimation aims to predict the translation quality of an MT system without relying on any reference. Given a source language sentence $X$ and a target language translation $T = \{t_1, t_2, \ldots, t_n\}$ with $n$ words, produced by an MT system, the QE system learns to predict two types of quality labels:

- **Word-level labels.** Label sequence $O = \{o_1, o_2, \ldots, o_n\}$, where $o_j$ is the quality label for the word translation $t_j$, which is usually a binary label meaning ‘OK’ or ‘BAD’;
- **Sentence-level label.** A quality score, $q$, for the whole translation $T$, e.g., Human-targeted Translation Edit Rate (HTER) score (Snover et al. 2006).

Commonly used QE datasets consist of machine translations, as well as the above labels, as tuples $\langle X, T, O, q \rangle$. This kind of label is expensive to get because it requires specific post-editing for the machine translations by experts with bilingual knowledge. Thus human-labeled QE datasets are usually limited in size.

**Predictor-Estimator Framework**

Since parallel data of the same translation direction is much more available, it is natural to transfer bilingual knowledge from parallel data to QE tasks. One well-known framework is the predictor-estimator framework (Kim, Lee, and Na 2017). It first trains a predictor on parallel data, which acts as a feature extractor. The estimator then uses the features from the predictor to make QE predictions. Here are two different kinds of predictors in previous research:

- **NMT-based Predictor.** Training a Neural Machine Translation (NMT) model on parallel data as the predictor (Zhou, Zhang, and Hu 2019) Fan et al. 2018 (Kim, Lee, and Na 2017).
- **PLM-based Predictor.** Using a Pretrained Language Model (PLM), e.g., BERT (Devlin et al. 2018), as the predictor Kepler et al. 2019 (Kim et al. 2019).

The task of both these predictor methods can be seen as a cross-lingual language model, which predicts unseen tokens given the source sentence $X$ and other tokens in the references $Y$. This task is different from the QE tasks, which aim to predict quality labels.

Once the predictor’s pretraining completes, an estimator is trained to incorporate the generation probability of each token in the translation $T$ as features (Fan et al.18) to predict word-level tags and sentence-level scores.

Please note that the estimator can only be trained on the labeled QE data, which is of limited size. On the other hand, there exist strong gaps between the predictor and estimator, preventing the model from making the best of parallel data.

**Approach**

To bridge the gaps in the data quality and training objectives between the original predictor and estimator, we propose the DirectQE framework. As shown in Figure 1, our framework includes a generator for pseudo QE data generation and a detector pretrained with the same QE objectives on the generated data.

**Generator**

The generator is trained on the parallel data to generate pseudo QE data. To achieve this goal, we train the generator as a word-level rewriter. The word-level rewriter is then used to produce pseudo translations with one-to-one correspondences with the references, from which labels could be automatically generated.

**Training the Word-level Rewriter on Parallel Data.** The generator adopts an encoder-decoder architecture like transformer (Vaswani et al. 2017). However, it uses a masked language model objective at the target side. Given a parallel sentence pair $\langle X, Y \rangle$, we generate a masked sentence $\hat{Y}$ by randomly replacing some tokens in the reference with a special tag [MASK]. The generator is trained to recover the masked tokens given the source tokens and the other tokens at the target side. Suppose the masked token is $y_j$, then the...
training objective $J_{\text{gen}}$ is to maximize

$$P(y_j | X, \tilde{Y}_{<j}, \text{[MASK]}, \tilde{Y}_{>j}; \theta).$$  

(1)

Following BERT (Devlin et al. 2018), we mask tokens with a 15% mask ratio.

We do not use a standard NMT model as the generator because it generates translations that may differ significantly from the reference ones. It is usually difficult to get trustworthy quality labels of these translations automatically.

**Generating Pseudo QE Translations.** We use the trained generator as a word rewriter to produce pseudo QE translations $Y' = \{y'_1, y'_2, \ldots, y'_n\}$. Notice that the length of $Y'$ is the same as the reference $Y$, and the tokens have one-to-one correspondences.

More specifically, we feed parallel data to the generator with some tokens at the target side masked, which is the same as the training process. For each masked token, the generator will output a probability distribution on the whole vocabulary. Unlike the common practice, we do not simply choose the best tokens with the highest generation probability. These tokens’ quality in sentences is often too good as a simulation for translation errors.

To generate the pseudo QE translations that share a closer data quality with real QE data than parallel data do, we choose to sample pseudo translations according to the probability distribution. Here, we introduce the following two different strategies:

- **Sample.** Similar to Clark et al. (2020), we sample tokens on the whole vocabulary according to the generation probability. To avoid mostly selecting the token with the highest probability, we sample tokens from a softmax with temperature $\tau$ (Hinton, Vinyals, and Dean 2015).

- **Top-k.** To improve sampling efficiency, we alternatively randomly select tokens from those with the top $k$ generation probability. The value of $k$ affects the overall quality of the translation sentence.

**Generating Pseudo QE Labels.** The generator produces a new translation $Y'$ for each parallel sentence pair. Then we generate pseudo QE labels on the translation for further training.

Word-level tags $O' = \{o'_1, o'_2, \ldots, o'_n\}$ are generated according to $Y'$ and $Y$, which marks a token as ‘BAD’ if it is generated and vice versa:

$$o'_j = \begin{cases} 1, & \text{if } y_j = y'_j, \\ 0, & \text{otherwise}. \end{cases}$$  

(2)

The sentence-level score $q'$ is generated as the ratio of generated tokens in a translation:

$$q' = 1 - \frac{\text{sum}(O')}{|\text{len}(O')}|.$$  

(3)

With the above QE labels, we can now get a pseudo tuple $\langle X, Y', O', q' \rangle$, which shares the same form of real QE data. While the real QE dataset is annotated by experts with bilingual knowledge and is quite limited in size, our methods can dynamically generate pseudo QE data at a scale even larger than the parallel data.

**Detector**

With the pseudo and real QE data, the detector could be pretrained and finetuned, respectively, using the same objectives. That is why we named our method “direct” pretraining.

**Pretraining on Pseudo QE Data.** The detector uses the pseudo QE data produced by the generator for pretraining. The pretraining task is to jointly predict the tags $O'$ at the word-level and score $q'$ at the sentence-level. The pretraining objective of word-level $J_w(X, Y', o'_j)$ is to maximize

$$\sum_{j=1}^{|O'|} \log P(o'_j | X, Y'; \theta),$$  

(4)

and that of sentence-level $J_s(X, Y', q')$ is to maximize

$$\log P(q' | X, Y'; \theta).$$  

(5)

The detector also uses the transformer as the basic framework, but it predicts the quality labels. It encodes the source sentence with self-attention to obtain hidden representations as the generator does. At the target side, the detector predicts word-level tags at each position with the representation from the last encoder layer. We also add a special tag $\text{[Ratio]}$ to the translation sequence, and its representation will be used to predict the sentence-level score.

Note that, the Sample and Topk strategies enable us to generate multiple translations given a parallel sentence pair. In practice, we generate new pseudo QE data for every mini-batch. Therefore, the detector could possibly be exposed to diverse translation errors, leading to better QE performance.

It is important to control the pretraining process for the detector. We randomly cut 2,000 sentence pairs out of the parallel data and use the generator to produce pseudo QE data with the Sample strategy. This dataset is used as the development set to monitor the pretraining process, and the performance on this dataset will be used for model selection.

**Finetuning on Real QE Data.** After pretraining, the detector will be finetuned on real QE tasks directly. The objectives of finetuning are $J_w(X, T, o_j)$ and $J_s(X, T, q)$ using real data and labels from QE datasets instead of pseudo ones.

When pretraining the detector, we do not finetune the generator together since it is difficult to back-propagate through the generator (Clark et al. 2020).

Unlike existing methods, with the help of the generator, we can pretrain the detector with QE objectives on a large amount of pseudo QE data, instead of training a language model task on parallel data. Furthermore, the detector can be directly used to do real QE tasks instead of introducing an extra estimator.

**Experiments**

**Experimental Settings**

**Dataset.** We carry out experiments on the WMT19 and WMT17 QE tasks for English-to-German (EN-DE) direction. The translation of the WMT19 task is generated by an NMT system, while in WMT17 a statistical MT system...
### Table 2: Main results of NMT-based QE, PLM-based QE, and our DirectQE on two EN-DE QE datasets.

| Sample       | Word-level Dev | Word-level Test | Sent-level Dev | Sent-level Test | F1-MULT |
|--------------|----------------|-----------------|----------------|-----------------|---------|
| DirectQE     |                |                 |                |                 |         |
| Top-k        | 57.99          | 15.72           | 57.19          | 15.72           | 40.63   |
| Sample       |                |                 |                |                 |         |
| DirectQE     | 73.38          | 14.72           | 73.56          | 13.15           | 58.18   |
| Top-k        |                |                 |                |                 |         |
| NMT-based    | 53.67          | 10.97           | 50.65          | 11.92           | 15.74   |
| PLM-based*   | 55.10          | 10.39           | 52.08          | 11.39           | 37.04   |
| DirectQE-Sample | 57.99 | 15.75          | 54.41          | 11.36           | 38.27   |
| DirectQE-Top-k | 57.19 | 15.72           | 55.08          | 11.25           | 40.63   |
| NMT-based    | 68.62          | 10.74           | 67.44          | 10.65           | 44.43   |
| PLM-based*   | 72.48          | 9.85            | 72.01          | 9.89            | 57.58   |
| DirectQE-Sample | 72.78 | 14.97          | 72.45          | 9.71            | 58.18   |
| DirectQE-Top-k | 73.38 | 14.72           | 73.56          | 9.51            | 57.99   |

### Models

- **NMT-based QE.** Using an NMT model as the predictor, and a Bi-LSTM (Graves and Schmidhuber 2005) as the estimator, following the architecture of QE Brain (Fan et al. 2018).
- **PLM-based QE.** Using BERT (Devlin et al. 2018) as the predictor, and a Bi-LSTM (Graves and Schmidhuber 2005) as the estimator. The code and pretrained parameters are from the huggingface (Wolf et al. 2019). Please note that BERT is pretrained with a large number of external resources.
- **DirectQE.** We implement the models described in the previous section with the Sample and Top-k strategies. When training the estimator, the model is jointly trained to predict the word-level and sentence-level QE labels.

### Implementation Details

- For NMT-based QE, the predictor of NMT-based QE consists of an encoder and a decoder, each of them is a 6-layer Transformer with hidden states dimension 512. The estimator is a single layer BiLSTM with hidden states dimension 512. The total number of parameters is about 113M.

  - For PLM-based QE, the encoder is the same as the released model, which is a 12-layer Transformer with hidden states dimension 768. The estimator is a single-layer Bi-LSTM with hidden states dimension 768. The total number of parameters is about 177M.

  - For DirectQE, the detector is based on the transformer (Vaswani et al. 2017), with one encoder and one decoder, also the same size as the NMT-based QE. The generator is based on a transformer of 6 layers but with hidden states dimension 256 for each layer (Clark et al. 2020). The total number of parameters is about 90M.

  Please note that our DirectQE is the smallest in size among all three architectures.

### Results of Ensemble Models

Instead of the single system results reported in the previous sub-section, many existing works of QE tasks only report their results of ensemble models. These results are hard to
compare because different research may use different ensemble methods or use different data. To get a relatively clear comparison, we only employ a very simple ensemble method: we perform a weighted summation on all the test results from our models with different settings and hyper-parameters, where the weights are learned from the development set. 

As shown in Table 3, our system can achieve better ensemble performance on both two datasets. We want to point out that the existing best ensemble system uses seven different methods, while our system only ensembles the three methods mentioned in our work. Since the DirectQE is conceptually distinct from the other two methods, and it is well known that diversity of models matters for ensemble, we expect our model could be beneficial for promoting the development of better QE systems.

Results on Other Language-pairs

We also conduct experiments on the WMT19 EN-RU and WMT20 EN-ZH QE tasks. For the EN-RU language pair, we use the official parallel dataset (12M) from the WMT19 QE task. For the EN-ZH language pair, we use the parallel dataset (7.5M) from the WMT18 translation task. As shown in Table 4, DirectQE can also achieve better performance on both datasets.

Analysis

We present analyses of our methods from two perspectives: data quality and training objectives.

Data Quality

Compared with parallel data, our generator produces translations with some errors, which is more similar to the quality estimation data. We first conduct experiments to show the effects of these data quality differences. 

Pseudo QE Data is Better than Parallel Data. Instead of using the detector, we keep the predictor-estimator architecture and train the NMT-based QE system. The performance is compared for the NMT-based systems trained with pseudo translations produced by our generator, and trained with the original parallel data. Table 5 shows the results. We can see that the performance is improved in most of the comparisons, demonstrating the benefits of generated pseudo QE data.

Quality of Pseudo Translations is Important. During the training of the generator, strategies and hyper-parameter choices may affect the quality of the generated pseudo translation. We present experiments to study these effects. We notice that the different choices of strategies and hyper-parameters mainly affect the ratio of replacement during the pseudo translation generation process. In the Sample strategy, the higher the temperature is, the more likely the current word is replaced. In the Top-k strategy, a hyper-parameter $k$ controls the replacement choices in generated pseudo translations, where higher $k$ values indicate a higher replacement ratio. 

Thus, we plot the performance of sentence-level quality estimation with the replacement ratio in Figure 3(a). As the replacement rate increases, the quality of pseudo-translations becomes worse and gradually approaches the real translation, so the QE performance first improves. However, when the replacement ratio is too high, the quality pseudo-translations may be worse than real translations which is clear harm to the QE results. In both strategies, there is a balance between pseudo data quality and QE results. The best performance is achieved at temperature 2 for
Figure 3: (a) QE performances according to different replacement ratios. The Top-k curve is models with k equals to 1, 3, 5, 10, 20, 40. The Sample curve is models with temperature equals to 1, 2, 4, 8. Rand is the model with random replacement. (b) The results on the pseudo QE development set according to the time steps.

| Dataset | Training  | Sent-level | Word-level |
|---------|-----------|------------|------------|
|         |           | Dev | Test     | Dev | Test     |
| WMT19   | parallel  | 53.67 | 50.65    | 15.74 | 17.08    |
|         | pseudo    | 54.11 | 51.50    | 18.85 | 16.69    |
| WMT17   | parallel  | 68.62 | 67.44    | 44.43 | -        |
|         | pseudo    | 69.45 | 68.19    | 45.99 | -        |

Table 5: Results of the the NMT-based QE system trained on the generated sentence pairs against those on the parallel sentence pairs.

the Sample strategy and top-10 for the Top-k strategy.

It could be easily seen that with a random replacement, the QE performance is quite weak (bottom-right corner) because the quality of the replacement cannot be ensured.

Diversity of Data Matters. During the pretraining of detector, we could re-sample from the generator distribution every epoch and generate the pseudo QE data dynamically. This enables us to train the detector with larger and more diverse data. To understand how this could affect the final performance, we use the generator to produce a fixed set of pseudo QE data, which is of the same size as the parallel data. We then compare the procedures of pretraining the detector on this fixed (static) set v.s. on the dynamically generated QE data.

As shown in Figure 3(b), the detector pretrained on the static set converges more quickly, but that pretrained on the dynamic data has better performance on the development set at last. Considering the gap of performance when the pretraining ends, it is obvious that the dynamic generated pseudo data is necessary for the detector.

Training Objective

Another gap in the predictor-estimator framework is the training objectives. We conduct the following experiments about DirectQE and the NMT-based system on the same pseudo QE data to investigate the influence of this gap.

The Detector Representation Contains More Information about the Current Tokens. Before making quality decisions (especially for word-level decisions), it is vital to gather enough information about the current tokens. We design experiments to measure the correlation between final representations and current tokens. For DirectQE, the final representations refer to the representations in each layer of the detector; for NLP-based systems, the final representations refer to the representations in each layer of the decoder part of the predictor. Our measurement is based on mutual information (MI) (Voita, Sennrich, and Titov 2019; Shwartz-Ziv and Tishby 2017). The bigger the MI value, the more information about current tokens in the final representation.

To estimate the MI, we follow the way described in Voita, Sennrich, and Titov (2019), which uses the clustering
Related Work

The combination of a generator and a detector looks like Generative Adversarial Network (GAN) (Caccia et al. 2018). However, we do not train these two parts together as GAN does. The usage of the generator is only to produce pseudo QE data for pretraining the detector.

Our method has a similar model architecture with the Electra (Clark et al. 2020). We both use a generator to produce pseudo sentences, and use a detector to find out the generated tokens in these sentences. However, we do not share the same motivation nor task. Their work improves on predictive pretraining for the masked language model, which lacks efficiency for general scenarios of natural language understanding, while a discrimination task may be more efficient. Our motivation is to bridge the gap between pretraining on parallel data and finetuning on QE data and build a direct pretraining method for QE tasks. Meanwhile, their model is monolingual and can not be used to solve QE tasks.

We also share the same idea of using pseudo data with some automatic post-editing (APE) work, which tries to use different MT systems to produce pseudo APE data (Negri et al. 2018; Junczys-Dowmunt and Grundkiewicz 2016). It is similar to what we do that we train the generator on parallel data to produce pseudo data. The difference is that they directly use MT systems to generate machine translations, while we modify ground-truth translations into corrupted but natural sentences, where the modified one-to-one corresponded tokens can be inferred to provide QE-oriented learning signals. That is, we require data that can be annotated automatically and trustworthy.

Conclusion

We propose a novel architecture called DirectQE that provides a direct pretraining for QE tasks. In our method, a generator is first trained on parallel data and will be used to produce the pseudo QE data. Then a detector will be directly pretrained with quality estimation on these pseudo data and then finetuned on real QE data. Compare with previous methods, our method can bridge the gaps in the data quality and training objectives between the pretraining and finetuning, which enables our model to learn more suitable knowledge for QE tasks from parallel data. Extensive experiments show the effects of each part of our method.

In future work, it is interesting to investigate automatically obtaining QE labels for naturally generated translations. Meanwhile, currently, we can only generate errors with substitution, and it is also interesting to simulate insertion and deletion operations and investigate their influences on translation quality.
Acknowledgements
We would like to thank the anonymous reviewers for their insightful comments. Shujian Huang is the corresponding author. This work is supported by National Key R&D Program of China (No. 2019QY1806), National Science Foundation of China (No. 61772261). This work is also supported by research funding from Huawei Corporation (No. YBN2020115023).

References
Abdelsalam, A.; Bojar, O.; and El-Beltagy, S. 2016. Bilingual Embeddings and Word Alignments for Translation Quality Estimation. In Proceedings of the First Conference on Machine Translation.

Caccia, M.; Caccia, L.; Fedus, W.; Larochelle, H.; Pineau, J.; and Charlin, L. 2018. Language GANs Falling Short. In Proceedings of the Seventh International Conference on Learning Representations.

Chen, Z.; Tan, Y.; Zhang, C.; Xiang, Q.; Zhang, L.; Li, M.; and Wang, M. 2017. Improving Machine Translation Quality Estimation with Neural Network Features. In Proceedings of the Second Conference on Machine Translation.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. International Conference on Learning Representations.

Denkowski, M.; and Lavie, A. 2014. Meteor Universal: Language-Specific Translation Evaluation for Any Target Language. In Proceedings of the Ninth Workshop on Statistical Machine Translation.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. 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.

Fan, K.; Wang, J.; Li, B.; Zhou, F.; Chen, B.; and Si, L. 2018. "Bilingual Expert" Can Find Translation Errors. In Proceedings of the AAAI Conference on Artificial Intelligence.

Graves, A.; and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. Advances in Neural Information Processing Systems.

Howard, J.; and Ruder, S. 2018. Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

Junczys-Dowmunt, M.; and Grundkiewicz, R. 2016. Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing. In Proceedings of the First Conference on Machine Translation.
Shwartz-Ziv, R.; and Tishby, N. 2017. Opening the Black Box of Deep Neural Networks via Information. *ArXiv* abs/1703.00810.

Snover, M.; Dorr, B.; Schwartz, R.; Micciulla, L.; and Makhoul, J. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems 30*.

Voita, E.; Sennrich, R.; and Titov, I. 2019. The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*.

Weiss, K.; Khoshgoftaar, T. M.; and Wang, D. 2016. A survey of transfer learning. *Journal of Big data*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2019. HuggingFace’s Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771.

Zhou, J.; Zhang, Z.; and Hu, Z. 2019. SOURCE: SOURce-Conditional Elmo-style Model for Machine Translation Quality Estimation. In *Proceedings of the Fourth Conference on Machine Translation*. 