Pushing the Limits of Translation Quality Estimation

André F. T. Martins
Unbabel
Instituto de Telecomunicações
Lisbon, Portugal
andre.martins@unbabel.com

Marcin Junczys-Dowmunt
Faculty of Mathematics and Computer Science
Adam Mickiewicz University in Poznań
junczys@amu.edu.pl

Fabio N. Kepler
Unbabel
L2F/INESC-ID, Lisbon, Portugal
University of Pampa, Alegrete, Brazil
fabio@kepler.pro.br

Ramón Astudillo
Unbabel
L2F/INESC-ID
Lisbon, Portugal
ramon@unbabel.com

Chris Hokamp
Dublin City University
Dublin, Ireland
chokamp@computing.dcu.ie

Abstract

Translation quality estimation is a task of growing importance in NLP, due to its potential to reduce post-editing human effort in disruptive ways. However, this potential is currently limited by the relatively low accuracy of existing systems. In this paper, we achieve remarkable improvements by exploiting synergies between the related tasks of word-level quality estimation and automatic post-editing. First, we stack a new, carefully engineered, neural model into a rich feature-based word-level quality estimation system. Then, we use the output of an automatic post-editing system as an extra feature, obtaining striking results on WMT16: a word-level F_1^{MULT} score of 57.47% (an absolute gain of +7.95% over the current state of the art), and a Pearson correlation score of 65.56% for sentence-level HTER prediction (an absolute gain of +13.36%).

1 Introduction

The goal of **quality estimation** (QE) is to evaluate a translation system's quality without access to reference translations (Blatz et al., 2004; Specia et al., 2013). This has many potential usages: informing an end user about the reliability of translated content; deciding if a translation is ready for publishing or if it requires human post-editing; highlighting the words that need to be changed. QE systems are particularly appealing for crowd-sourced and professional translation services, due to their potential to dramatically reduce post-editing times and to save labor costs (Specia, 2011). The increasing interest on this problem from an industrial angle comes as no surprise (Turchi et al., 2014; de Souza et al., 2015; Martins et al., 2016; Kozlova et al., 2016).

In this paper, we tackle **word-level QE**, whose goal is to assign a label of OK or BAD to each word in the translation (Figure 1). Past approaches to this problem include linear classifiers with handcrafted features (Ueffing and Ney, 2007; Biçici, 2013; Shah et al., 2013; Luong et al., 2014), sometimes subject to feature selection (Avramidis, 2012; Beck et al., 2013), recurrent neural networks (de Souza et al., 2014; Kim and Lee, 2016), and systems that combine linear and neural models (Kreutzer et al., 2015; Martins et al., 2016). We start by proposing a "pure" QE system (§3) consisting of a new, carefully engineered, neural model (NEURALQE), stacked into a linear feature-rich classifier (LINEARQE). Along the way, we provide a rigorous empirical analysis to better understand the contribution of the several groups of features and to justify the architecture of the neural system.

A second contribution of this paper is bringing in the related task of **automatic post-editing** (APE; Simard et al. (2007)), which aims to automatically

Source	*The Sharpen tool sharpens areas in an image .*
MT	*Der Schärfen-Werkezug* Bereiche in einem Bild *schärfer erscheint .*
PE (reference)	Mit dem Scharfzeichner können Sie einzelne Bereiche in einem Bild scharfzeichnen .
QE	*BAD BAD* OK OK OK OK *BAD BAD* OK ‖ HTER = 66.7%

Figure 1: Example from the WMT16 word-level QE training set. Shown are the English source sentence, the German translation, its manual post-edition, and the conversion to word quality labels made with the TERCOM tool (`http://www.cs.umd.edu/~snover/tercom`). Words labeled as OK are shown in green, and those labeled as BAD are shown in *red*. We also show the HTER (fraction of edit operations to produce PE from MT) computed by TERCOM.

correct the output of machine translation (MT). We show that a variant of the APE system of Junczys-Dowmunt and Grundkiewicz (2016), trained on a large amount of artificial "roundtrip translations," is extremely effective when adapted to predict word-level quality labels (yielding APEQE, §4). We further show that the pure and the APE-driven QE systems are highly complementary (§5): a stacked combination of LINEARQE, NEURALQE, and APEQE boosts the scores even further, leading to a new state of the art in the WMT16 dataset: an F_1^{MULT} score of 57.47%, which represents an absolute improvement of +7.95% over the previous best system.

Finally, we provide a simple word-to-sentence conversion to adapt our system to **sentence-level QE**. This results in a new state of the art for human-targeted translation error rate (HTER) prediction, where we obtain a Pearson's r correlation score of 65.56% (+13.36% absolute gain), and for sentence ranking, which achieves a Spearman's ρ correlation score of 65.92% (+17.62%). We complement our findings with error analysis that highlights the synergies between pure and APE-driven QE systems.

2 Datasets and System Architecture

Datasets. For developing and evaluating our systems, we use the datasets listed in Table 1. These datasets have been used in the QE and APE tasks in WMT 2015–2016 (Bojar et al., 2015, 2016).[1] They span two language pairs (English-Spanish and English-German) and two different domains (news translations and information technology). We used the standard train, development and test splits. Each split contains the source and automatically translated sentences (which we use as inputs), the manually post-edited sentences (output for the APE task), and a sequence of OK/BAD quality labels, one per each translated word (output for the word-level QE task); see Figure 1. Besides these datasets, for training the APE system we make use of artificial roundtrip translations; this will be detailed in §4.

Evaluation. For all experiments, we report the official evaluation metrics of each dataset's year. For WMT15, the official metric for the word-level QE task is the F_1 score of the BAD labels (F_1^{BAD}). For WMT16, it is the product of the F_1 scores for the OK and BAD labels (denoted F_1^{MULT}). For sentence-level QE, we report the Pearson's r correlation for HTER prediction and the Spearman's ρ correlation score for sentence ranking (Graham, 2015).

From post-edited sentences to quality labels. In the datasets above, the word quality labels are obtained automatically by aligning the translated and the post-edited sentence with the TERCOM software tool (Snover et al., 2006), with the default settings (tokenized, case insensitive, exact matching only, shifts disabled). This tool computes the HTER (the normalized edit distance) between the translated and post-edited sentence. As a by-product, it aligns the words in both sentences, identifying substitution errors, word deletions (i.e. words omitted by the translation system), and insertions (redundant words in the translation). The substitution errors and the insertions originate the BAD quality labels.

The fact that the quality labels are automatically obtained from the post-edited sentences is not just an artifact of these datasets, but a procedure that is highly convenient for developing QE systems in an industrial setting. Manually annotating word-level quality labels is time-consuming and expensive; on the other hand, post-editing translated sentences is commonly part of the workflow of crowd-sourced and professional translation services. Thus, getting

[1] Publicly available at `http://www.statmt.org/wmt15` and `http://www.statmt.org/wmt16`.

Dataset	Language pair	# sents	# words
WMT15, Train	En-Es	11,271	257,548
WMT15, Dev	En-Es	1,000	23,207
WMT15, Test	En-Es	1,817	40,899
WMT16, Train	En-De	12,000	210,958
WMT16, Dev	En-De	1,000	19,487
WMT16, Test	En-De	2,000	34,531

Table 1: Datasets used in this paper.

quality labels for free from sentences that have already been post-edited is a much more realistic and sustainable process. This observation suggests that we can tackle word-level QE in two ways:

1. **Pure QE:** run the TER alignment tool (i.e. TERCOM) on the post-edited data, and then train a QE system directly on the generated quality labels;
2. **APE-based QE:** train an APE system on the original post-edited data, and at runtime use the TER aligment tool to convert the automatically post-edited sentences to quality labels.

From a machine learning pespective, QE is a sequence labeling problem (i.e., whose output sequence has a fixed length and a small number of labels), while APE is a sequence-to-sequence problem (where the output is of variable length and spans a large vocabulary). Therefore, we can regard APE-based QE as a "projection" of a more complex and fine-grained output into a simpler output space. APE-based QE systems have the potential for being more powerful since they are trained with this finer-grained information (provided there is enough training data to make them generalize well). We report results in §4 confirming this hypothesis.

Our system architecture, described in full detail in the following sections, consists of state-of-the-art pure QE and APE-based QE systems, which are then combined to yield a new, more powerful, QE system.

3 Pure Quality Estimation

The best performing system in the WMT16 word-level QE task was developed by the Unbabel team (Martins et al., 2016). It is a pure but rather complex QE system, ensembling a linear feature-based classifier with three different neural networks with different configurations. In this section, we provide a simpler version of their system, by replacing the three ensembled neural components by a single one, which we engineer in a principled way. We evaluate the resulting system on additional data (WMT15 in addition to WMT16), covering a new language pair and a new content type. Overall, we obtain a slightly higher accuracy with a much simpler system.

In this section, we describe the linear (§3.1) and neural (§3.2) components of our system, as well as their combination (§3.3).

3.1 Linear Sequential Model

We start with the linear component of our model, a discriminative feature-based sequential model (called LINEARQE), based on Martins et al. (2016). The system receives as input a tuple $\langle s, t, \mathcal{A} \rangle$, where $s = s_1 \dots s_M$ is the source sentence, $t = t_1 \dots t_N$ is the translated sentence, and $\mathcal{A} \subseteq \{(m, n) \mid 1 \leq m \leq M,\ 1 \leq n \leq N\}$ is a set of word alignments. It predicts as output a sequence $\widehat{y} = y_1 \dots y_N$, with each $y_i \in \{\textsc{bad}, \textsc{ok}\}$. This is done as follows:

$$\begin{aligned} \widehat{y} \;&=\; \arg\max_{y} \sum_{i=1}^{N} \boldsymbol{w}^\top \boldsymbol{\phi}_u(s, t, \mathcal{A}, y_i) \\ &\quad + \sum_{i=1}^{N+1} \boldsymbol{w}^\top \boldsymbol{\phi}_b(s, t, \mathcal{A}, y_i, y_{i-1}). \qquad (1) \end{aligned}$$

Above, $\boldsymbol{w}$ is a vector of weights, $\boldsymbol{\phi}_u(s, t, \mathcal{A}, y_i)$ are unigram features (depending only on a single output label), $\boldsymbol{\phi}_b(s, t, \mathcal{A}, y_i, y_{i-1})$ are bigram features (depending on consecutive output labels), and y_0 and y_{N+1} are special start/stop symbols.

Features. Table 2 shows the unigram and bigram features used in the LINEARQE system. Like the baseline systems provided in WMT15/16, we include features that depend on the target word and its aligned source word, as well as the context surrounding them.[2] A distinctive aspect of our system is the inclusion of syntactic features, which were found useful to detect grammatically incorrect constructions, as we shall see.[3] We use features

[2]Features involving the aligned source word are replaced by NIL if the target word is unaligned. If there are multiple aligned source words, they are concatenated into a single feature.

[3]While syntactic features have been used previously in sentence-level QE (Rubino et al., 2012), they have never been applied to the finer-grained word-level variant tackled here.

that involve the dependency relation, the head word, and second-order sibling and grandparent structures. Features involving part-of-speech (POS) tags and syntactic information are obtained with TurboTagger and TurboParser (Martins et al., 2013).[4]

Training. The feature weights are learned by running 50 epochs of the max-loss MIRA algorithm (Crammer et al., 2006), with regularization constant $C \in \{10^{-k}\}_{k=1}^{4}$ and a Hamming cost function placing a higher penalty on false positives than on false negatives ($c_{\mathrm{FP}} \in \{0.5, 0.55, \ldots, 0.95\}$, $c_{\mathrm{FN}} = 1 - c_{\mathrm{FP}}$), to account for the existence of fewer BAD labels than OK labels in the data. These values are tuned on the development set.

Results and feature contribution. Table 3 shows the results. To help understand the contribution of each group of features, we evaluated different variants of the LINEARQE system on the development sets of WMT15/16. As expected, the use of bigrams improves the simple unigram model, and the syntactic features help even further. The impact of these features is more prominent in WMT16: the rich bigram features lead to scores about 3 points above a sequential model with a single indicator bigram feature, and the syntactic features contribute another 2.5 points. The net improvement exceeds 6 points over the unigram model.

3.2 Neural System

Next, we describe the neural component of our pure QE system, which we call NEURALQE. In WMT15 and WMT16, the neural QUETCH system (Kreutzer et al., 2015) and its ensemble with other neural models (Martins et al., 2016) were components of the winning systems. However, none of these neural models managed to outperform a linear model when considered in isolation—for example, QUETCH obtained a F_1^{BAD} of 35.27% in the WMT15 test set, far below the 40.84% score of the linear system built by the same team. By contrast, our carefully engineered NEURALQE model attains a performance superior to that of the linear system, as we shall see.

Architecture. The architecture of NEURALQE is depicted in Figure 2. We used Keras (Chollet, 2015) to implement our model. The system receives as

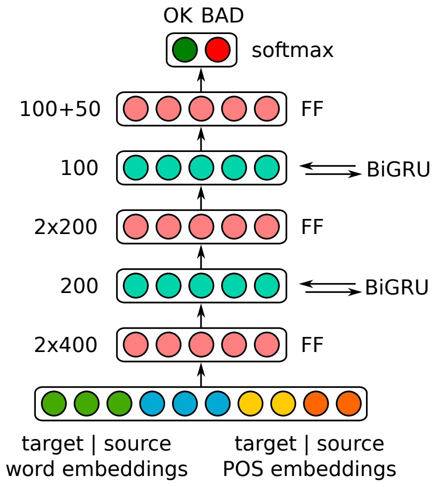

Figure 2: Architecture of our NEURALQE system.

input the source and target sentences s and t, their word-level alignments $\mathcal{A}$, and their corresponding POS tags obtained from TurboTagger. The input layer follows a similar architecture as QUETCH, with the addition of POS features. A vector representing each target word is obtained by concatenating the embedding of that word with those of the aligned word in the source.[5] The immediate left and right contexts for source and target words are also concatenated. We use pre-trained 64-dimensional Polyglot word embeddings (Al-Rfou et al., 2013) for English, German, and Spanish; these word embeddings are then refined during training. In addition to this, POS tags for each source and target word are also embedded and concatenated. POS embeddings have size 50 and are initialized as described by Glorot and Bengio (2010). A dropout probability of 0.5 is applied to the resulting vector representations.

The following layers are then applied in sequence:

1. Two feed-forward layers of size 400 with rectified linear units (ReLU; Nair and Hinton (2010));
2. A bidirectional GRU layer of size 200, where forward and backward vectors are concatenated, trained with layer normalization (Ba et al., 2016);
3. Two feed-forward ReLU layers of size 200;
4. A bidirectional GRU layer of size 100 with identical configuration to the previous GRU;

[4]http://www.cs.cmu.edu/~ark/TurboParser.

[5]For the cases in which there are multiple source words aligned to the same target word, the embeddings are averaged.

Features	Label	Input (referenced by the ith target word)
unigram	$y_i \wedge \dots$	*BIAS *WORD, LEFTWORD, RIGHTWORD *SOURCEWORD, SOURCELEFTWORD, SOURCERIGHTWORD *LARGESTNGRAMLEFT/RIGHT, SOURCELARGESTNGRAMLEFT/RIGHT *POSTAG, SOURCEPOSTAG †WORD+LEFTWORD, WORD+RIGHTWORD †WORD+SOURCEWORD, POSTAG+SOURCEPOSTAG
simple bigram	$y_i \wedge y_{i-1} \wedge \dots$	*BIAS
rich bigrams	$y_i \wedge y_{i-1} \wedge \dots$ $y_{i+1} \wedge y_i \wedge \dots$	all above WORD+SOURCEWORD, POSTAG+SOURCEPOSTAG
syntactic	$y_i \wedge \dots$	DEPREL, WORD+DEPREL HEADWORD/POSTAG+WORD/POSTAG LEFTSIBWORD/POSTAG+WORD/POSTAG RIGHTSIBWORD/POSTAG+WORD/POSTAG GRANDWORD/POSTAG+HEADWORD/POSTAG+WORD/POSTAG

Table 2: Features used in the LINEARQE system (see Martins et al., 2016 for a detailed description). Features marked with * are included in the WMT16 baseline system. Those marked with † were proposed by Kreutzer et al. (2015).

Features	WMT15 (F_1^{BAD})	WMT16 (F_1^{MULT})
unigrams only	41.77	40.05
+simple bigram	42.20	40.63
+rich bigrams	42.80	43.65
+syntactic (full)	**43.68**	**46.11**

Table 3: Performance on the WMT15 (En-Es) and WMT16 (En-De) development sets of several configurations of LINEARQE. We report the oficial metric for these shared tasks, F_1^{BAD} for WMT15 and F_1^{MULT} for WMT16.

5. Two more feed-forward ReLU layers of sizes 100 and 50, respectively.

As the output layer, a softmax transformation over the OK/BAD labels is applied. The choice for this architecture was dictated by experiments on the WMT16 development data, as we next explain.

Training. We train the model with the RMSProp algorithm (Tieleman and Hinton, 2012) by minimizing the cross-entropy with a linear penalty for BAD word predictions, as in Kreutzer et al. (2015). We set the BAD weight factor to 3.0. All hyperparameters are adjusted based on the development set.

Results and architectural choices. The final results are shown in Table 4. Overall, the final NEURALQE model achieves an F_1^{MULT} score of 46.80% on the WMT16 development set, compared with the 46.11% obtained with the LINEARQE system (cf. Table 3). This contrasts with previous neural systems, such as QUETCH (Kreutzer et al., 2015) and any of the three neural systems developed by Martins et al. (2016), which could not outperform a rich feature linear classifier.

To justify the most relevant choices regarding the architecture of NEURALQE, we also evaluated several variations of it on the WMT16 development set. The use of recurrent layers yields the largest contribution to the performance of NEURALQE, as the scores drop sharply (by more than 4 points) if they are replaced by feed-forward layers (which would correspond to a mere deeper QUETCH model). The first RNN is particulary effective, as scores drop more than 2 points if it is removed. The use of layer normalization on the recurrent layers also contributes positively (+1.20) to the final score. As expected, the use of POS tags adds another large improvement: everything staying the same, the model without POS tags as input performs almost 2.5 points worse. Finally, varying the size of the hidden layers and the depth of the networks hurts the final model's performance, albeit more slightly.

Model	F_1^{MULT}
NEURALQE	**46.80**
No POS tags	44.41 (-2.39)
Replace RNN by FF	42.36 (-4.44)
Only the first RNN	45.76 (-1.04)
Only the second RNN	44.37 (-2.43)
Remove FF between RNNs	46.35 (-0.45)
Narrower layers	45.09 (-1.71)
Broader layers	45.02 (-1.78)
One more layer at the end	46.31 (-0.49)
No layer normalization	45.60 (-1.20)

Table 4: Effect of architectural changes in NEURALQE on the WMT16 development set.

Model	F_1^{BAD} dev	F_1^{BAD} test
Best system in WMT15	43.1-	43.12
QUETCH+ (2nd best)	–	43.05
LINEARQE	43.68	42.50
NEURALQE	43.51	43.35
NEURALQE-5	44.21	43.54
NEURALQE-15	44.11	**43.93**
STACKEDQE	**44.68**	43.70

Table 5: Performance of the pure QE systems on the WMT15 datasets. The best performing system in the WMT15 competition was by Esplà-Gomis et al. (2015), followed by Kreutzer et al. (2015)'s QUETCH+, which is also an ensemble of a linear and a neural system.

3.3 Stacking Neural and Linear Models

We now stack the NEURALQE system (§3.2) into the LINEARQE system (§3.1) as an ensemble strategy; we call the resulting system STACKEDQE.

Stacking architectures (Wolpert, 1992; Breiman, 1996) have proved effective in structured NLP problems (Cohen and de Carvalho, 2005; Martins et al., 2008). The underlying idea is to combine two systems by letting the prediction of the first system be used as an input feature for the second system. During training, it is necessary to jackknife the first system's predictions to avoid overfitting the training set. This is done by splitting the training set in K folds (we set $K = 10$) and training K different instances of the first system, where each instance is trained on $K - 1$ folds and makes predictions for the left-out fold. The concatenation of all the predictions yields an unbiased training set for the second classifier.

Neural intra-ensembles. We also evaluate the performance of intra-ensembled neural systems. We train independent instances of NEURALQE with different random initializations and different data shuffles, following the approach of Jean et al. (2015) in neural MT. In Tables 5–6, we report the performance on the WMT15 and WMT16 datasets of systems ensembling 5 and 15 of these instances, called respectively NEURALQE-5 and NEURALQE-15. The instances are ensembled by taking the averaged probability of each word being BAD. We see consistent benefits (both for WMT15 and WMT16) in ensembling 5 neural systems and (somewhat surprisingly) some degradation with ensembles of 15.

Stacking architecture. The individual instances of the neural systems are incorporated in the stacking architecture as different features, yielding STACKEDQE. In total, we have 15 predictions (probability values given by each NEURALQE system) for every word in the training, development and test datasets. These predictions are plugged as additional features in the LINEARQE model. As unigram features, we used one real-valued feature for every model prediction at each position, conjoined with the label. As bigram features, we used two real-valued features for every model prediction at the two positions, conjoined with the label pair.

The results obtained with this stacked architecture on the WMT15 and WMT16 datasets are shown respectively in Tables 5-6. In WMT15, it is unclear if stacking helps over the best intra-ensembled neural system, with a slight improvement in the development set, but a degradation in the test set. In WMT16, however, stacking is clearly beneficial, with a boost of about 2 points over the best intra-ensembled neural system and 3–4 points above the linear system, both in the development and test partitions. For the remainder of this paper, we will take STACKEDQE as our pure QE system.

4 APE-Based Quality Estimation

Now that we have described a pure QE system, we move on to an APE-based QE system (APEQE).

Our starting point is the system submitted by the Adam Mickiewicz University (AMU) team to the APE task of WMT16 (Junczys-Dowmunt and Grundkiewicz, 2016). They explored the applica-

Model	F_1^{MULT} dev	F_1^{MULT} test
Best system in WMT16	**49.25**	49.52
Unbabel-Linear (2nd best)	45.94	46.29
LINEARQE	46.11	46.16
NEURALQE	46.80	47.29
NEURALQE-5	47.30	48.50
NEURALQE-15	46.77	47.98
STACKEDQE	49.16	**50.27**

Table 6: Performance of the pure QE systems on the WMT16 datasets. The best performing system in the WMT16 competition was by Martins et al. (2016). We show also the performance of the linear system developed by the same team (Unbabel-Linear).

tion of neural translation models to the APE problem and achieved good results by treating different models as components in a log-linear model, allowing for multiple inputs (the source s and the translated sentence t) that were decoded to the same target language (post-edited translation p). Two systems were considered, one using s as the input (SRC→PE) and another using t as the input (MT→PE). A simple string-matching penalty integrated within the log-linear model was used to control for higher faithfulness with regard to the raw MT output. The penalty fires if the APE system proposes a word in its output that has not been seen in t.

To overcome the problem of too little training data, Junczys-Dowmunt and Grundkiewicz (2016) generated large amounts of artificial data via **round-trip translations**: a large corpus of monolingual sentences is first gathered for the target language in the domain of interest (each sentence is regarded as an artificial post-edited sentence p); then an MT system is ran to translate these sentences to the source language (which are regarded as the source sentences s), and another MT system in the reverse direction translates the latter back to the target language (playing the role of the translations t). The artificial data has been filtered to match the HTER statistics of the training and development data for the shared task.[6] Their submission improved over the uncorrected baseline on the unseen WMT16 test set by -3.2% TER and +5.5% BLEU and outperformed any other system submitted to the shared-task by a large margin.

[6]The artificial filtered data has been made available by the authors at `https://github.com/emjotde/amunmt/wiki/AmuNMT-for-Automatic-Post-Editing`.

4.1 Training the APE System

We reproduce the experiments from Junczys-Dowmunt and Grundkiewicz (2016) using Nematus (Sennrich et al., 2016) for training and AmuNMT (Junczys-Dowmunt et al., 2016) for decoding.

As stated in §3.3, jackknifing is required to avoid overfitting during the training procedure of the stacked classifiers (§5), therefore we start with preparing four jackknifed models. We perform the following steps:

- We divide the original WMT16 training set into four equally sized parts, maintaining correspondences between different languages. Four new training sets are created by leaving out each part and concatenating the remaining three parts.
- For each of the four new training sets, we train one APE model on a concatenation of the smaller set of artificial data (denoted as "round-trip.n1" in Junczys-Dowmunt and Grundkiewicz (2016), consisting of 531,839 sentence triples) and a 20-fold oversampled new training set. Each of these newly created four APE models has not seen a different part of the quartered original training data.
- To avoid overfitting, we use scaling dropout[7] over GRU steps and input embeddings, with dropout probabilities 0.2, and over source and target words with probabilities 0.1 (Sennrich et al., 2016).
- We use Adam (Kingma and Ba, 2014) instead of Adadelta (Zeiler, 2012).
- We train both models (SRC→PE and MT→PE) until convergence up to 20 epochs, saving model checkpoints every 10,000 mini-batches.
- The last four model checkpoints of each training run are averaged element-wise (Junczys-Dowmunt et al., 2016) resulting in new single models with generally improved performance.

To verify the quality of the APE system, we ensemble the 8 resulting models (4 SRC→PE and 4 MT→PE) and add the APE penalty described in Junczys-Dowmunt and Grundkiewicz (2016). This

[7]Currently available in the MRT branch of Nematus at `https://github.com/rsennrich/nematus`

System	TER↓	BLEU↑
Best system at WMT16	**21.52**	**67.65**
Uncorrected MT (baseline)	24.76	62.11
APE MT→PE	22.60	66.50
APE SRC→PE	28.39	57.90
APE TER-tuned	**20.99**	**67.62**

Table 7: Results on the official WMT16 test set for the APE task, in comparison to Junczys-Dowunt and Grundkiewicz (2016). Bold results mark comparable systems.

	F_1^{MULT} dev	F_1^{MULT} test
APE MT→PE	27.46	31.39
APE SRC→PE	51.92	53.70
APE TER-tuned	40.17	41.87
APE QE-tuned (APEQE)	**54.95**	**55.68**

Table 8: Performance of APE-based QE systems in the WMT16 development and test sets.

large ensemble across folds is only used during test time. For creating the jackknifed training data, only the models from the corresponding fold are used. Since we combine models of different types, we tune weights on the development set with MERT[8] (Och, 2003) towards TER and achieve the results listed in Table 7 for the APE shared task. For the purely SRC→PE and MT→PE ensembles, models are weighted equally. We achieve slightly better results in terms of TER, the main task metric, than the original system using less data.

4.2 Adaptation to QE and Task-Specific Tuning

As described in §2, APE outputs can be turned into word quality labels using TER-based word alignments. Somewhat surprisingly, among the previously introduced APE systems, we observe (Table 8) that the SRC→PE APE system is the so-far strongest stand-alone QE system in this work. This system is essentially a retrained neural MT component without any additional features.[9] The MT→PE system and the TER-tuned APE ensemble are much weaker in terms of F_1^{MULT}. However, we can obtain an even better APE-based QE system by retuning the full APE ensemble towards F_1^{MULT}, the official QE metric.[10] With this approach, we produce our new best stand-alone QE-system, which we call APEQE. This system is about 2 points ahead of the APE SRC→PE system (see Table 8).

5 Full Stacked System

Finally, we consider a larger stacked system where we stack both NEURALQE and APEQE into LINEARQE. This will mix pure QE with APE-based QE systems; we call the result FULLSTACKEDQE. The procedure is analogous to that described in §3.3, with one extra binary feature for the APE-based word quality label predictions. For training, we used jackknifing as described in §3.3.

5.1 Word-Level QE

The performance of the FULLSTACKEDQE system on the WMT16 dataset is shown in Table 9. We compare with the other systems introduced in this paper, and with the best participating system at WMT16 (Martins et al., 2016).

We can see that the APE-driven and the pure QE systems are complementary: the full combination of the linear, neural, and APE-based systems improves the scores another 2 points with respect to the best individual system (APEQE). Overall, we obtain an F_1^{MULT} score of 57.47%, a new state of the art and an absolute gain of +7.95% over Martins et al. (2016). This is a remarkable improvement that can pave the way for a wider adoption of word-level QE systems in industrial settings. In §6, we analyze the errors made by the pure and APE-based QE systems to better understand how they complement each other.

5.2 Sentence-Level QE

Encouraged by the strong results obtained with the FULLSTACKEDQE system in word-level QE, we investigate how we can adapt this system for HTER prediction at sentence level. Prior work (de Souza et al., 2014) incorporated word-level quality predictions as features in a sentence-level QE system, training a feature-based linear classifier. Here, we

[8]We found MERT to work better when tuning towards TER than kb-MIRA which has been used in the original paper.

[9]Note that this system resembles other QE approaches which use pseudo-reference features (Albrecht and Hwa, 2008; Soricut and Narsale, 2012; Shah et al., 2013), since the SRC→PE is essentially an "alternative" MT system.

[10]Using again MERT and executing 7 iterations on the official development set with an n-best list size of 12.

	F_1^{MULT} dev	F_1^{MULT} test
Best system in WMT16	49.25	49.52
LINEARQE	46.11	46.16
NEURALQE	46.80	47.29
STACKEDQE	49.16	50.27
APEQE	54.95	55.68
FULLSTACKEDQE	**56.80**	**57.47**

Table 9: Performance of the several word-level QE systems on the WMT16 development and test datasets. The baseline is the best participating system in WMT16, from the Unbabel team (Martins et al., 2016).

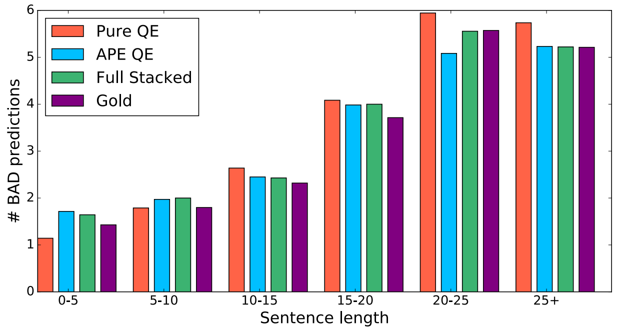

Figure 3: Averaged number of words predicted as BAD by the different systems and in the gold dev set, for different bins of the sentence length.

show that a very simple conversion, which requires no training or tuning, is enough to obtain a substantial improvement over the state of the art.

For the APE system, it is easy to obtain a prediction for HTER: we can simply measure the HTER between the translated sentence t and the predicted corrected sentence $\widehat{p}$. For a pure QE system, we apply the following word-to-sentence conversion technique: (i) run a QE system to obtain a sequence of OK and BAD word quality labels; (ii) use the fraction of BAD labels as an estimate for HTER. Note that this procedure, while not requiring any training, is far from perfect: words that are not in the translated sentence but exist in the reference post-edited sentence do not originate BAD labels, and therefore will not contribute to the HTER estimate. Yet, as we will see, this procedure applied to the STACKEDQE system (i.e. without the APEQE component) is already sufficient to obtain state of the art results. Finally, to combine the APE and pure QE systems toward sentence-level QE, we simply take the average of the two HTER predictions above.

Table 10 shows the results obtained with our pure QE system (STACKEDQE), with our APE-based system (APEQE), and with the combination of the two (FULLSTACKEDQE). As baselines, we report the performance of the two best systems in the sentence-level QE task at WMT16 (Kozlova et al., 2016; Kim and Lee, 2016).

The results are striking: even our weakest system (STACKEDQE) with the simple conversion procedure above is already sufficient to obtain state of the art results, outperforming Kozlova et al. (2016) and Kim and Lee (2016) by a considerable margin. The APEQE system gives a very large boost over these scores, which are further increased by the combined FULLSTACKEDQE system. Overall, we obtain absolute gains of +13.36% in Pearson's r correlation score for HTER prediction, and +17.62% in Spearman's ρ correlation for sentence ranking, a considerable advance over the previous state of the art.

6 Error Analysis

Performance over sentence length. To better understand the differences in performance between the pure QE system (STACKEDQE) and the APE-based system (APEQE), we analyze how the two systems, as well as their combination (FULLSTACKEDQE), perform as a function of the sentence length.

Figure 3 shows the averaged number of BAD predictions made by the three systems for different sentences lengths, in the development set. For comparison, we show also the true average number of BAD words in the gold standard. We observe that, for short sentences (less than 5 words), the pure QE system tends to be too optimistic (i.e., it underpredicts BAD words) and the APE-based system too pessimistic (overpredicting them). In the range 5-10 words, the pure QE system matches the proportion of BAD words more accurately than the APE-based system. For medium/long sentences, we observe the opposite behavior (this is particularly clear in the 20-25 word range), with the APE-based system being generally better. On the other hand, the combination of the two systems (FULLSTACKEDQE) manages to find a good balance between these two biases, being much closer to the true proportion of BAD labels for

	Pearson dev	Pearson test	Spearman dev	Spearman test
Best system in WMT16 (HTER prediction)	–	52.5-	–	–
Best system in WMT16 (ranking)	–	46.0-	–	48.3-
StackedQE	55.30	54.93	56.46	55.34
ApeQE	59.04	61.27	61.06	62.48
FullStackedQE	**64.04**	**65.56**	**65.52**	**65.92**

Table 10: Performance of our sentence-level QE systems on the WMT16 datasets. The baselines are the best WMT16 system in the HTER prediction track (Kozlova et al., 2016) and in the sentence ranking track (Kim and Lee, 2016).

Source	*Combines the hue value of the blend color with the luminance and saturation of the base color to create the result color .*
MT	Kombiniert den Farbton Wert der Angleichungsfarbe mit der Luminanz und Sättigung der Grundfarbe zu erstellen .
PE (Reference)	Kombiniert den Farbtonwert der Mischfarbe mit der Luminanz und Sättigung der Grundfarbe .
APE	Kombiniert den Farbton der Mischfarbe mit der Luminanz und die Sättigung der Grundfarbe , um die Ergebnisfarbe zu erstellen .
StackedQE	Kombiniert *den Farbton Wert* der *Angleichungsfarbe* mit der Luminanz und Sättigung der Grundfarbe *zu erstellen* .
ApeQE	Kombiniert den Farbton *Wert* der *Angleichungsfarbe* mit der Luminanz und Sättigung der Grundfarbe zu erstellen .
FullStackedQE	Kombiniert den *Farbton Wert* der *Angleichungsfarbe* mit der Luminanz und Sättigung der Grundfarbe *zu erstellen* .
Source	*The Video Preview plug - in supports RGB , grayscale , and indexed images .*
MT	Mit dem Zusatzmodul “ Videovorschau ” unterstützt RGB- , Graustufen- und indizierte Bilder .
PE (Reference)	Das Zusatzmodul “ Videovorschau ” unterstützt RGB- , Graustufen- und indizierte Bilder .
APE	Das Dialogfeld “ Videovorschau ” unterstützt RGB- , Graustufen- und indizierte Bilder .
StackedQE	*Mit dem* Zusatzmodul “ Videovorschau ” *unterstützt RGB-* , Graustufen- und indizierte Bilder .
ApeQE	*Mit dem Zusatzmodul* “ Videovorschau ” unterstützt RGB- , Graustufen- und indizierte Bilder .
FullStackedQE	*Mit dem* Zusatzmodul “ Videovorschau ” unterstützt RGB- , Graustufen- und indizierte Bilder .

Table 11: Examples on WMT16 validation data. For each, we show the source and translated sentences, the gold post-edited sentence, the output of the APE system, and the QE predictions of our pure QE system, our APE-based QE system, and their combination as the full stacked system. Words predicted as OK are shown in green, and those predicted as BAD are shown in *red*. For both examples, the full stacked system predicts all quality labels correctly.

both shorter and longer sentences than any of the individual systems. This shows that the two systems complement each other well in the combination.

Illustrative examples. Table 11 shows concrete examples of quality predictions on the WMT16 development data. In the top example, we can see that the APE system correctly replaced *Angleichungsfarbe* by *Mischfarbe*, but is under-corrective in other parts; the APEQE system therefore misses several BAD words, but manages to get the correct label (OK) for *den*. By contrast, the pure QE system erroneously flags this word as incorrect, but it makes the right decision on *Farbton* and *zu erstellen*, being more accurate than APEQE. The combination of the two systems (pure QE and APEQE) leads to the correct sequential prediction. In the bottom example, the pure QE system assigns the correct label to *Zusatzmodul*, while the APE system mistranslates this word to *Dialogfeld*, leading to a wrong prediction by the APEQE system. On the other hand, pure QE misclassifies *unterstützt RGB-* as BAD words, while the APEQE gets them right. Overall, the APEQE is more accurate in this example. Again, these decisions complement each other well, as can be seen by the combined QE system which outputs the correct word labels for the entire sentence.

7 Conclusions

We have presented new state-of-the-art systems for word-level and sentence-level QE, that are considerably more accurate than previously existing systems on the WMT16 dataset.

First, we proposed a new pure QE system which stacks a linear and a neural system, and is simpler and slighly more accurate than the currently best word-level system. Then, by relating the tasks of APE and word-level QE, we derived a new APE-based QE system, which leverages additional artificial roundtrip translation data, achieving a larger improvement. Finally, we combined the two systems via a full stacking architecture, boosting the scores even further. Error analysis shows that the pure and APE-based systems are highly complementary. The full system was extended to sentence-level QE by virtue of a simple word-to-sentence conversion, requiring no further training or tuning.

Acknowledgments

This work was partially supported by the the EXPERT project (EU Marie Curie ITN No. 317471), and by Fundação para a Ciência e Tecnologia (FCT), through contracts UID/EEA/50008/2013 and UID/CEC/50021/2013, the LearnBig project (PTDC/EEI-SII/7092/2014), and the GoLocal project (grant CMUPERI/TIC/0046/2014).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. *arXiv preprint arXiv:1307.1662*.

Joshua Albrecht and Rebecca Hwa. 2008. The role of pseudo references in MT evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 187–190.

Eleftherios Avramidis. 2012. Quality estimation for machine translation output using linguistic analysis and decoding features. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 84–90.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Daniel Beck, Kashif Shah, Trevor Cohn, and Lucia Specia. 2013. SHEF-Lite: When less is more for translation quality estimation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 335–340.

Ergun Biçici. 2013. Referential translation machines for quality estimation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 343–351.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proc. of the International Conference on Computational Linguistics*, page 315.

Ondrej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, , Philipp Koehn, , Christof Monz, Matteo Negri, Pavel Pecina, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Tenth Workshop on Statistical Machine Translation*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198.

Leo Breiman. 1996. Stacked regressions. *Machine Learning*, 24:49.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

William W. Cohen and Vitor R. de Carvalho. 2005. Stacked sequential learning. In *Proc. of International Joint Conference on Artificial Intelligence*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

José G. C. de Souza, Jesús González-Rubio, Christian Buck, Marco Turchi, and Matteo Negri. 2014. FBK-UPV-UEdin participation in the WMT14 Quality Estimation shared-task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 322–328.

José G. C. de Souza, Marcello Federico, and Hassan Sawaf. 2015. MT Quality Estimation for E-Commerce Data. In *Proc. of MT Summit XV, vol. 2: MT Users' Track*, pages 20–29.

Miquel Esplà-Gomis, Felipe Sánchez-Martínez, and Mikel Forcada. 2015. UAlacant word-level machine translation quality estimation system at WMT 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 309–315, September.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Yvette Graham. 2015. Improving evaluation of machine translation quality estimation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, pages 1804–1813.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for wmt15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 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*, pages 751–758.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? A case study on 30 translation directions. *arXiv preprint arXiv:1610.01108*.

Hyun Kim and Jong-Hyeok Lee. 2016. Recurrent neural network based translation quality estimation. In *Proc. of the First Conference on Machine Translation*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Anna Kozlova, Mariya Shmatova, and Anton Frolov. 2016. YSDA Participation in the WMT'16 Quality Estimation Shared Task. In *Proceedings of the First Conference on Machine Translation*, pages 793–799.

Julia Kreutzer, 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.

Ngoc Quang Luong, Laurent Besacier, and Benjamin Lecouteux. 2014. LIG System for Word Level QE task at WMT14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 335–341.

André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking Dependency Parsers. In *Proc. of Empirical Methods for Natural Language Processing*.

André F. T Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

André F. T. Martins, Ramón Astudillo, Chris Hokamp, and Fábio Kepler. 2016. Unbabel's participation in the wmt16 word-level translation quality estimation shared task. In *Proceedings of the First Conference on Machine Translation*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proc. of the International Conference on Machine Learning*, pages 807–814.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the Annual Meeting on Association for Computational Linguistics*, pages 160–167.

Raphael Rubino, Jennifer Foster, Joachim Wagner, Johann Roturier, Rasul Samad Zadeh Kaljahi, and Fred Hollowood. 2012. DCU-Symantec submission for the WMT 2012 quality estimation task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 138–144.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376.

Kashif Shah, Trevor Cohn, and Lucia Specia. 2013. An investigation on the effectiveness of features for translation quality estimation. In *Proceedings of the Machine Translation Summit*, volume 14, pages 167–174.

Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. 2007. Rule-based translation with statistical phrase-based post-editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231.

Radu Soricut and Sushant Narsale. 2012. Combining quality prediction and system selection for improved automatic translation output. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 163–170.

Lucia Specia, Kashif Shah, Jose G.C. de Souza, and Trevor Cohn. 2013. QuEst - a translation quality estimation framework. In *Proc. of the Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84.

Lucia Specia. 2011. Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of the 15th Conference of the European Association for Machine Translation*, pages 73–80.

Tijmen Tieleman and Geoffrey Hinton. 2012. Rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2).

Marco Turchi, Antonios Anastasopoulos, José GC de Souza, and Matteo Negri. 2014. Adaptive quality estimation for machine translation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, pages 710–720.

Nicola Ueffing and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40.

D. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–260.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.