Unbabel's Participation in the WMT16 Word-Level Translation Quality Estimation Shared Task

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

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

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

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

Abstract

This paper presents the contribution of the Unbabel team to the WMT 2016 Shared Task on Word-Level Translation Quality Estimation. We describe our two submitted systems: (i) UNBABEL-LINEAR, a feature-rich sequential linear model with syntactic features, and (ii) UNBABEL-ENSEMBLE, a stacked combination of the linear system with three different deep neural networks, mixing feed-forward, convolutional, and recurrent layers. Our systems achieved $F_1^{\text{OK}} \times F_1^{\text{BAD}}$ scores of 46.29% and 49.52%, respectively, which were the two highest scores in the challenge.

1 Introduction

Quality estimation is the task of evaluating a translation system's quality without access to reference translations (Specia et al., 2013; Bojar et al., 2015). This paper describes the contribution of the Unbabel team to the Shared Task on Word-Level Quality Estimation (QE Task 2) at the 2016 Conference on Statistical Machine Translation (WMT 2016). The task aims to predict the word-level quality of English-to-German machine translated text, by assigning a label of OK or BAD to each word in the translation.

Our system's architecture is inspired by the recent QUETCH+ system (Kreutzer et al., 2015), which achieved top performance in the WMT 2015 Word Level QE task (Bojar et al., 2015). QUETCH+ predicts the labels of individual words by combining a linear feature-based classifier with a feedforward neural network (called QUETCH, for QUality Estimation from scraTCH). The linear classifier is based upon Luong et al. (2014) and uses the baseline features provided in the shared task. The QUETCH neural network is a multi-layer perceptron, which takes as input the embeddings of the target words and the aligned source words, along with their context, and outputs a binary label for the target word. The combination is done by stacking the scores of the neural network and the linear classifier as additional features in another linear classifier.

Our main contributions are the following:

- We replaced the word-level linear classifier in QUETCH+ by a sentence-level first-order sequential model. Our model incorporates rich features for label unigrams and bigrams, detailed in §2.1–2.2.
- We included syntactic features that look at second-order dependencies between target words. This is explained in §2.3.
- We implemented three different neural systems, one extension of the original QUETCH model and two recurrent models with different depth. These are detailed in §3.1–3.3.
- We ensembled multiple versions of each neural system for different data shuffles and initializations as additional features for the linear system, via a stacking architecture. This is detailed in §4.

The following external resources were used: part-of-speech tags and extra syntactic dependency information were obtained with TurboTagger and TurboParser (Martins et al., 2013),[1]

[1]Publicly available on http://www.cs.cmu.edu/~ark/TurboParser/.

Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 806–811,
Berlin, Germany, August 11-12, 2016.

trained on the Penn Treebank (for English) and on the version of the German TIGER corpus used in the SPMRL shared task (Seddah et al., 2014). For the neural models, we used pre-trained word embeddings from Polyglot (Al-Rfou et al., 2013) and embeddings obtained from a trained neural MT system (Bahdanau et al., 2014).

2 Linear Sequential Model

Our starting point is a discriminative feature-based linear sequential model. The input is a tuple $x := \langle \boldsymbol{s}, \boldsymbol{t}, \mathcal{A} \rangle$, where $\boldsymbol{s} = s_1 \dots s_M$ is the source sentence, $\boldsymbol{t} = t_1 \dots t_N$ 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. The goal is to predict a label sequence $\widehat{\boldsymbol{y}} = y_1, \dots, y_N$, where each $y_i \in \{\text{BAD}, \text{OK}\}$. This is done as follows:

$$\widehat{\boldsymbol{y}} = \arg\max_{y} \sum_{i=1}^{N} \boldsymbol{w} \cdot \boldsymbol{f}_u(x, y_i) + \sum_{i=1}^{N+1} \boldsymbol{w} \cdot \boldsymbol{f}_b(x, y_i, y_{i-1}), \quad (1)$$

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

A detailed description of the features used in our submitted systems is provided below. The weights for these features are learned by running 50 epochs of the max-loss MIRA algorithm (Crammer et al., 2006) with $C = 0.001$. The cost function takes into account mismatches between predicted and gold labels, with a higher cost on false positives ($c_{FP} = 0.8$) and a lower cost on false negatives ($c_{FN} = 0.2$), to compensate for the fact that there are fewer BAD labels than OK labels in the data. These values were tuned on the development set.

2.1 Unigram Features

We used the following unigram features, taken from the baseline features provided by the organizers (with some slight changes that are detailed below). Each of the features below is conjoined with the target label at each position.

- BIAS. A bias feature.
- WORD, LEFTWORD, RIGHTWORD. Lexical features for the target word in the current, previous, and next positions.
- SOURCEWORD, SOURCELEFTWORD, SOURCERIGHTWORD. Lexical features for the source word aligned to the current target word, and their left/right neighboring words in the source sentence; these will all be NULL if the target word is unaligned. If there are multiple aligned source words, they are all concatenated into a single SOURCEWORD feature, and the contextual features are with respect to the leftmost and rightmost aligned source words, respectively.[2]
- LARGESTNGRAMLEFT/RIGHT, SOURCELARGESTNGRAMLEFT/RIGHT. The language model features provided by the shared task organizers, containing the length of the largest n-gram on each direction observed in the target and source language models.[3]
- POSTAG, SOURCEPOSTAG. Part-of-speech tag of the current target word, and of the source-aligned word, both predicted by TurboTagger (Martins et al., 2013). The latter will be NULL if the target word is unaligned, and a concatenation of POS tags if there are multiple aligned source words.[4]

Following Kreutzer et al. (2015), we conjoined some of the baseline features above as follows.

- WORD+LEFTWORD, WORD+RIGHTWORD. Bilexical features including the target word in the current position, conjoined with the previous/next target word.
- WORD+SOURCEWORD, POSTAG+SOURCEPOSTAG. Features conjoining the source and target word/POS tag.

2.2 Bigram Features

We constructed rich bigram features which conjoin the label pair (for each pair of consecutive target words) with two copies of the features in §2.1: one copy for the first word in the pair, and another for the second word in the pair. Furthermore, we also introduced the following trilexical features:

- WORDPAIR+LEFTWORD, WORDPAIR+RIGHTWORD. Trilexical features

[2] This is slightly different from the baseline feature provided by the organizers of the shared task, which consider the single source word aligned with the highest confidence.

[3] We did not use the provided backoff language model features.

[4] This differs from the features provided by the organizers in two ways: the POS tagger is different; and the SOURCEPOSTAG can have multiple tags for many-to-one alignments.

including the two target words in the current position, conjoined with the target word in the previous and next positions, respectively.

To understand the importance of these enriched feature sets, we also tried a "simple bigram" model using only a single indicator feature for the label pair.

2.3 Syntactic Features

A major novelty in our quality estimation system is the usage of syntactic features, which are useful to detect grammatically incorrect constructions. We used the following syntactic features based on dependencies predicted by TurboParser (Martins et al., 2013). With the exception of the first one, all the syntactic features below were used for unigrams only.

- DEPREL, WORD+DEPREL. The dependency relation between the current target word and its parent, as well as its conjunction with the target word itself.
- HEADWORD/POSTAG+WORD/POSTAG. Conjunction of the word/POS tag of the current target word with the one of its syntactic head.
- LEFTSIBWORD/POSTAG+WORD/POSTAG, RIGHTSIBWORD/POSTAG+WORD/POSTAG. Same, but for the closest sibling in the left/right.
- GRANDWORD/POSTAG+HEADWORD/POSTAG +WORD/POSTAG. Up to bilexical features involving the grandparent, head, and current target word/POS tag, including backed off versions of these features.

2.4 Performance of the Linear System

To help understand the contribution of each group of features in §2.1–2.3, we evaluated the performance of different variants of the UNBABEL-LINEAR system on the development set.

The results are shown in Table 1. As expected, the use of bigrams improves the simple unigram model, which is similar to the baseline model provided by the organizers. We can also see that the rich bigram features have a great impact in the scores (about 2.6 points above a sequential model with a single indicator bigram feature), and that the syntactic features help even further, contributing another 2.6 points. The net improvement exceeds 6.5 points over the unigram model.

Features	$F_1^{\text{GOOD}} \times F_1^{\text{BAD}}$
unigrams only	39.27
+simple bigram	40.65
+rich bigrams	43.31
+syntactic	45.94

Table 1: Performance on the dev-set of several configurations of the UNBABEL-LINEAR system. The model with simple bigrams has a single BIAS bigram feature, conjoined with the label pairs.

3 Neural Models

We next describe the three neural models implemented by our team. Five instances of each model were trained using different data shuffles, following the idea of Jean et al. (2015), and also using random initialization seeds (except for the model in §3.1). These instances were incorporated to the stacking architecture as different features, as will be described in §4. For reporting purposes, the instances of each model were also ensembled with two strategies: majority voting, where each instance contributes one vote, and averaged probability, where we average each instance's predicted probability of a word being BAD).

3.1 Feedforward Network

Our feedforward network is an adaptation of QUETCH (Kreutzer et al., 2015). The model classifies each word as OK/BAD by using each target language word and the corresponding aligned word from the source language as input. To increase the context available to the model, the words at the left and right of each target and source word are provided as well. Each of the 6 words is represented by a pre-trained word embedding, and all are concatenated into a single vector. There is a single hidden layer, which uses a hyperbolic tangent (`tanh`) non-linearity. The model is trained using stochastic gradient descent to maximize the log-likelihood. During training, the loss function is weighted to penalize BAD instances in order to compensate for the asymmetry in OK/BAD labels in the corpus.

All hyperparameters were tuned on the development set. The best performance was attained by using 64-dimensional Polyglot embeddings (Al-Rfou et al., 2013), updated during training, and a hidden layer size of 20. Words with no pre-trained embeddings were initialized to a vector of zeros and optimized during training. The BAD weight

was set to 3. Despite its simplicity, this model provided a good performance when compared to neural models using no extra features.

In order to adapt to this year's shared task, several improvements were introduced to QUETCH. Similarly to the linear model, for many-to-one alignments we included all aligned words in the source (not just ones), with their corresponding size one contexts (see footnote 2). To obtain a fixed size vector, the average over the embeddings of each aligned word was used. A second improvement was the addition of a convolutional layer spanning 3 words after the hidden layer. This aimed to expand the local context used by QUETCH. Finally, a dropout of 20% after the concatenation of the embeddings was applied. Of the implemented improvements, dropout had the largest effect, whereas including all aligned words brought a small but consistent improvement.

When the five trained instances were ensembled by average probability, this rather simple approach led to a large improvement in performance, as shown in Table 2.

Model	$F_1^{\text{OK}} \times F_1^{\text{BAD}}$
Single Feed-forward Network (FFN)	41.74
5 FFN Ensemble (Majority Voting)	43.26
5 FFN Ensemble (Average Probability)	43.47

Table 2: Effect of intra-model ensembling of the feed-forward network reproducing QUETCH.

3.2 Bilingual, Bidirectional Recurrent Model

We also implemented a bidirectional model which takes target words and their aligned source words as inputs, and outputs a OK/BAD tag for each target word. The internal representations of the bidirectional model are then passed to a feedforward network where the first layer performs a maxout transformation with two pieces (Goodfellow et al., 2013). Dropout is applied before multiplication with the maxout layer's weights. The maxout layer is followed by two fully-connected layers with 100 and 50 hidden units, respectively. `tanh` is used as the non-linear function between layers. During development, we validated that this model improved performance over a vanilla feed-forward network using the WMT 2015 English-to-Spanish dataset.

Source and target word embeddings are initialized with embeddings from an English-German neural machine translation (NMT) system (Bahdanau et al., 2014), trained with all data from the WMT 2015 English-to-German translation task (Bojar et al., 2015). The vocabulary size is 100,000 for both English and German, and words in the training data which do not occur in the NMT vocabulary are mapped to an "unknown" token. We experimented with tuning the embedding parameters during training, but found that leaving them static led to better performance.

Gated Recurrent Units (GRU) are used for the recurrent transitions (Chung et al., 2015). The size of the hidden layers is fixed at 500, and the embedding size is set to 200. Minibatch size is fixed at 40. Dropout is applied to all feedforward parameters of the models, but not to the parameters of the recurrent transitions. We tested the impact of ℓ_2 regularization, and our best performing system uses both dropout and ℓ_2 regularization.[5] All recurrent models are optimized using AdaDelta (Zeiler, 2012). The best model for each training run was selected using early-stopping according to the F_1-product score of the model on the development set. The intra-model ensembling results are shown in Table 3.

Model	$F_1^{\text{OK}} \times F_1^{\text{BAD}}$
Single Model (BBRM)	40.95
5 BBRN (Majority Voting)	41.52
5 BBRN (Average Probability)	41.31

Table 3: Effect of intra-model ensembling of the Bilingual, Bidirectional Recurrent Model.

3.3 Multi-Feature Convolutional Recurrent Network

Our second recurrent model uses both recurrent and convolutional components, along with POS tags obtained from TurboTagger for both target and aligned source words. As in §3.2, an entire sentence is fed at once to the network, which takes as input the target words and the aligned source words, both with their respective left and right contexts, and the POS tags for each target and source words, both with the two left and two right tags (i.e., we use a convolution with a window of size 3 for words and 5 for POS tags). The output is a sequence of OK/BAD tags for the target words.

The first network's layer are embeddings for all the aforementioned inputs: word embeddings are

[5] Peak performance was obtained with dropout probability set to 0.5, and ℓ_2 regularization coefficient $\alpha = 10^{-4}$.

initialized with Polyglot embeddings, as in §3.1, and tag embeddings of size 50 are initialized randomly. All are further trained along with the other network parameters. For each input timestep, all embeddings are concatenated and then passed to two consecutive feedforward hidden layers with 200 units. A bidirectional GRU layer with 200 units is then applied across all timesteps. The resulting representations are further passed to another feedforward network consisting of two layers of 200 units, followed by a softmax layer which classifies a target word as OK or BAD.

All activations besides softmax are rectified linear units (unlike the models in §3.1–§3.2, which use `tanh` activations), and a dropout of 20% is used in each layer. Optimization is carried out by RMSProp.[6] As in §3.2, early-stopping based on the F_1-product score over the development set was used for selecting the best model of each training run.

We verified empirically that shallower models performed worse, while the new POS tags and specially the middle bidirectional GRU gave a boost in score.

Model	$F_1^{\mathbf{OK}} \times F_1^{\mathbf{BAD}}$
Single Model (MFCRN)	44.33
5 MFCRN (Majority Voting)	46.58
5 MFCRN (Average Probability)	46.10

Table 4: Effect of intra-model ensembling of the multi-feature convolutional recurrent network model.

Table 4 shows the average performance of five trained instances and the ensembles performances of these instances as described in §3, which also led to large improvements as in the other models.

4 Stacking Architecture

As described in §3, each of the three neural models produced five trained instances, yielding 15 predictions in total for every word in the training, development and test datasets. For the three models, we used 10-fold jackknifing to obtain unbiased predictions for the training set. We then plugged these 15 predictions (as probability values) as additional features in the linear model described in §2. 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.

[6] T. Tieleman and G. Hinton, unpublished.

	$F_1^{\mathbf{OK}} \times F_1^{\mathbf{BAD}}$
Linear + 5 FFN	46.89
Linear + 5 BBLM	47.01
Linear + 5 MFCRN	48.58
Full Ensemble	49.25

Table 5: Performance of a stacked network ensembling each of the three deep models and the linear model, and of a full ensemble (UNBABEL-ENSEMBLE).

	F_1^{OK}	F_1^{BAD}	$F_1^{\text{OK}} \times F_1^{\text{BAD}}$
UNBABEL-LINEAR	87.48	52.92	46.29
UNBABEL-ENSEMBLE	88.45	55.99	49.52

Table 6: Performance of the submitted systems on the test set.

The results obtained with this stacked architecture are shown in Table 5, where we compare with smaller ensembles that stack each individual deep model with the linear one (using only 5 extra features instead of 15). We can see that there is a clear benefit in combining all the deep models, which suggests that these systems complement each other by focusing on different quality aspects.

5 Final Results

Finally, we show in Table 6 the results obtained in the test set for our two submitted systems, UNBABEL-LINEAR and UNBABEL-ENSEMBLE. As expected, the ensemble system gave an additional boost (>3 points) over the linear model, which is consistent with the findings of the previous sections on the validation data.

6 Conclusions

We have presented a novel linear sequential model which uses the baseline task features along with a new set of syntactic features, leading to top performance on the word-level quality estimation task. Using this model as our baseline, we obtain further improvements by including a version of the feedforward QUETCH system, as well as two novel recurrent models, as stacked features in the sequential linear model. Our final ensemble achieved the best performance of all submitted systems.

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

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

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

[Bojar et al.2015] 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*, WMT, Lisbon, Portugal.

[Chung et al.2015] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2067–2075.

[Crammer et al.2006] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

[Goodfellow et al.2013] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. 2013. Maxout networks. *CoRR*, abs/1302.4389.

[Jean et al.2015] 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.

[Kreutzer et al.2015] 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.

[Luong et al.2014] 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, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

[Martins et al.2013] 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*.

[Seddah et al.2014] Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109.

[Specia et al.2013] Lucia Specia, Kashif Shah, Jose G.C. 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, Sofia, Bulgaria, August. Association for Computational Linguistics.

[Zeiler2012] Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701.