Data Enhancement and Selection Strategies for the Word-level Quality Estimation

Varvara Logacheva[§], **Chris Hokamp**[†], **Lucia Specia**[§]
§Department of Computer Science, University of Sheffield, UK
{v.logacheva,l.specia}@sheffield.ac.uk

†CNGL Centre for Global Intelligent Content
Dublin City University, Ireland
chokamp@computing.dcu.ie

Abstract

This paper describes the DCU-SHEFF word-level Quality Estimation (QE) system submitted to the QE shared task at WMT15. Starting from a baseline set of features and a CRF algorithm to learn a sequence tagging model, we propose improvements in two ways: (i) by filtering out the training sentences containing too few errors, and (ii) by adding incomplete sequences to the training data to enrich the model with new information. We also experiment with considering the task as a classification problem, and report results using a subset of the features with Random Forest classifiers.

1 Introduction

The WMT shared task on Quality estimation (QE) for Machine Translation (MT) has included the sub-task on the QE at the word level since the year 2013. The goal of this task is to assign a quality label to each word of an automatically translated sentence without using its reference translations. The set of possible output labels can vary. Labels can specify the edit action which should be performed on the word in order to improve the sentence (substitution, deletion, insertion) — these labels were used in the WMT13 QE task (Bojar et al., 2013). Labels can be further refined to specify the type of error: grammar error, wrong terminology, untranslated word, etc., motivated by the MQM error typology [1] — this tagging was used in last year's task (Bojar et al., 2014). In both cases, tags can be generalised to a binary label, "GOOD" or "BAD", indicating whether or not the word is correct.

This year, the word-level QE task (Task 2 in WMT15 QE shared task[2]) consists in assigning only a binary label ("GOOD" or "BAD") to every word in automatically translated sentences — that is, to identify if a word is suitable for this sentence or should be modified. The possible errors are substitution (word replacement) or insertion. This formulation of the task cannot detect deletions in the MT hypothesis, because there is a one-to-one correspondence between tokens in the hypothesis and output tags.

The data for the word-level QE task was produced for one translation direction, namely from English into Spanish. The training, development and test datasets have been translated automatically with an online statistical MT system, and then post-edited by human translators. Besides the datasets themselves, baseline feature sets were provided. The suggested baseline training model is conditional random fields (CRF) (Lafferty et al., 2001), which is one of the most widely used techniques for sequence labelling. The baseline tagging for this task was done with CRF model trained using CRF++ tool[3].

Our system uses the baseline features released for the task and the same tool which was used for baseline model generation. However, we performed data selection and bootstrapping techniques that led to significant improvement over the baseline.

2 Baseline setting

The goal of the system was to estimate the quality of machine-translated sentences at the word-level, i.e. to assign every word a label "GOOD" or "BAD" depending on its quality. Therefore, the training and test data contains the following information: the source sentences, their automatic

[1] http://www.qt21.eu/launchpad/content/multidimensional-quality-metrics

[2] http://www.statmt.org/wmt15/quality-estimation-task.html

[3] https://code.google.com/p/crfpp/

Proceedings of the Tenth Workshop on Statistical Machine Translation, pages 330–335,
Lisboa, Portugal, 17-18 September 2015.

translations into the target language, the manual post-editions (corrections) of the automatic translations, and the word-level tags for the automatic translations.

The tags were acquired by aligning the machine translations with their post-editions using the TER tool (Snover et al., 2006). Unchanged words were assigned the label "GOOD", words which were substituted with another word or deleted by a post-editor were assigned the label "BAD". The "BAD" labels thus correspond to the "addition" and "substitution" edit operations in the word-level string alignment between the MT hypothesis and the post-edited segment.

The dataset contains automatic translations from English into Spanish. The training data consists of 11,271 sentences, the development and test sets have 1,000 and 1,817 sentences, respectively. The post-editions and tags for the test data were not made available until after the end of the evaluation period.

2.1 Features

We used a subset of features described by Luong et al. (2014), mainly the features that were listed as the most informative. This corresponds to the baseline feature set released for the shared task. The full list of features is the following:

- Word count features:
  - source and target token counts
  - source and target token count ratio
- Lexical features:
  - target token
  - target token's left and right contexts of 1 word
- Alignment features:
  - source word aligned to the target token
  - source word's left and right contexts of 1 word
- Boolean dictionary features:
  - target token is a stopword
  - target token is a punctuation mark
  - target token is a proper noun
  - target token is a number
- Target language model features:
  - order of the highest order ngram which ends with the target token
  - order of the highest order ngram which starts with the target token
  - backoff behaviour of the ngram (t_{i-2}, t_{i-1}, t_i), where t_i is the target token (backoff behaviour is computed as described in Raybaud et al. (2011))
  - backoff behavior of the ngram (t_{i-1}, t_i, t_{i+1})
  - backoff behavior of the ngram (t_i, t_{i+1}, t_{i+2})
- Source language model features:
  - order of the highest order ngram which ends with the source token
  - order of the highest order ngram which starts with the source token
- Boolean pseudo-reference feature: 1 if the token is contained in the pseudo-reference, 0 otherwise[4]
- Part-of-speech features[5]:
  - POS of the target token
  - POS of the source token
- WordNet features:
  - Number of senses for the target token
  - Number of senses for the source token

2.2 Alternative system

We performed additional experiments with a reduced feature set which does not contain lexical and alignment features. These features were excluded in order to enable the use of classifiers implemented in the `scikit-learn`[6] toolkit. The implementations in this toolkit can only deal with scalar features directly. Therefore, in order to use categorical features (e.g. strings), these need to be converted into one-hot vector representation.

The one-hot representation of a categorical feature is the representation of every possible feature

[4]The pseudo-reference used for this feature extraction is the automatic translation generated by an English-Spanish phrase-based statistical MT system trained on the Europarl corpus (Koehn, 2005) using Moses system with standard settings (`http://www.statmt.org/moses/?n=Moses.Baseline`).

[5]POS tagging was performed with TreeTagger tool `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`

[6]`http://scikit-learn.org/`

value from a domain $\mathbf{D}$ as a vector of 0s and a single 1. The length of such vector is $|D|$ (length of the set of possible values of the feature), every position in the vector corresponds to a value from *D*. Each instance of this feature should correspond to a vector which has only one element with value 1 at the position of the categorical value taken by this instance of the feature. Since the categorical features used rely on a very large vocabulary, converting them into one-hot vectors would have increased the feature space significantly, resulting in very sparse feature vectors.

Systems using shorter feature sets (i.e. without lexical and alignment features) were trained with the Random Forest classifier in `scikit-learn` with default settings. This scenario considers each (feature vector, token, tag) tuple as a separate instance, so that we no longer explicitly model the dependencies in the sequence. However, contextual information about the token is still included in the feature set via several other features (see Section 2.1), so sequence information is not completely disregarded in this scenario.

2.3 Baseline results

The baseline results for our systems on the development set are outlined in Table 1. Since instances of the "GOOD" class are much more numerous than instances tagged as "BAD", the average F1-score is dominated by the F1-GOOD. However, the F1-GOOD is high for any system, as even a naive system tagging all words as "GOOD" would score high. This metric is thus uninformative. Therefore, the primary quality metric for this task is F1-BAD. The performance of the Random Forest classifier is significantly higher than that of CRF model, although it uses a smaller feature set and does not take the labelling context into account.

	F1-BAD	F1-GOOD	Weighted F1
Baseline (CRF)	0.18	0.88	0.75
Reduced (Random Forest)	0.24	0.86	0.78

Table 1: Baseline results.

The scores given here and further in the paper are for the development set, as this dataset was used for tuning the systems and choosing the settings to be submitted for the task. The scores for the test set on the official submissions are given in Section 5. These are a bit lower, but they maintain the relative trend (i.e. the systems that perform better on the development set perform better on the test set as well).

3 Generating Data by Bootstrapping New Examples

Although the size of training data is considerably larger than the size of datasets that have been used before, it may still be too sparse to perform QE at the word level. This is because not all tokens are shared between the training and test datasets. Instead of using a data selection method to choose training examples which correspond to the dev/test sets, we decided to enhance the training data with additional samples generated from the initial dataset. This corresponds more closely with a realistic deployment scenario for a word-level QE system, where the test set is unknown.

We tested two methods of additional data generation:

- In addition to every complete sentence from the training data we used sequences that consist of the first n words of this sentence, where $n \in [1, N]$ (N = number of words in the sentence). For example, for each sentence of 10 words we added nine new training examples: a sequence that consists of the first word only, a sequence that consists of the first two words, the first three words, etc. This strategy is further referred to as **1-to-N**.

- For every sentence from the training data we used all trigrams of this sentence as training examples. This strategy will be denoted as **ngram**.

Another idea is to perform bootstrapping not only to expand the training data, but also to break the test set into smaller chunks for tagging.

Bootstrapping for the test set is produced as follows. In order to tag a sequence $\mathbf{s} = s_1 s_2 \ldots s_n$ we convert it into a list of n sub-sequences $L_\mathbf{s} = [s_1;\ s_1 s_2;\ s_1 s_2 s_3;\ \ldots;\ s_1 s_2 \ldots s_n]$. Each sub-sequence from $L_\mathbf{s}$ is tagged by the system. The final tagging for every word $s_i \in \mathbf{s}$ is taken from a sub-sequence where s_i is the last symbol, so that we compose the final tagging for the sequence **s** from the tags for words s_i listed in bold: $[\mathbf{s_1};\ s_1\mathbf{s_2};\ s_1 s_2\mathbf{s_3};\ \ldots;\ s_1 s_2 \ldots \mathbf{s_n}]$.

The described scenario refers to the **1-to-N** bootstrapping method for the test set. The **ngram**

bootstrapping method for the test set can be used analogously.

The intuition behind this approach is the following. If we train a system on a set of incomplete sequences (1-to-N or ngrams), it might capture local dependencies which do not hold for complete sentences. Therefore, in order to improve the prediction accuracy we should test the system on incomplete sequences as well. There are many possibilities for combining the partial sequence predictions (e.g. averaging the scores of one word in different incomplete sequences or training a linear regression model to find a weight for every prediction), but in this experiment we tested only one strategy: taking the score of the *i*-th word from the *i*-th sequence.

	Training	plain	1-to-N	ngram
	Test ↓			
CRF	plain	0.170	0.238	0.213
	1-to-N	0.221	**0.251**	0.212
	ngram	0.170	0.238	0.226
Random Forest	plain	0.236		0.239
	1-to-N	**0.255**		0.237
	ngram	0.234		**0.255**

Table 2: Experiments with bootstrapped data (F1-score for "BAD" class). 'plain' setting means no bootstrapping (original data).

We tested all the training and test data bootstrapping techniques. The results are outlined in Table 2. We used three different training sets: the original dataset with no bootstrapping (denoted as 'plain' in the table), a dataset bootstrapped with the **1-to-N** strategy, and one bootstrapped with the **ngram** strategy, and three different test sets (analogously, plain, 1-to-N, and ngram). We trained two systems for every combination of datasets: one system performs sequence labelling with CRF, the other classifies words with a Random Forest classifier. That would give us $3 \times 3 \times 2 = 18$ systems. However, the experiments with training data enhanced with **1-to-N** strategy could not be performed for Random Forest classifier due to computational complexity, so we are effectively comparing 15 combinations of labelling strategies and bootstrapping techniques.

The CRF model benefits from both strategies: when bootstrapping only training data the F1-score increases from 0.17 to 0.21 (**ngram**) and 0.23 (**1-to-N**). Bootstrapping of test data brings an additional improvement: even when the training set is not changed, applying **1-to-N** strategy to the test increases the score from 0.17 to 0.22. However, **ngram** bootstrapping of the test proved ineffective unless it was applied to the training data as well.

We assume that bootstrapping the training data helps due to the fact that in the CRF model all instances within a sequence are influenced by each other: the choice of tag for a word is dependent on all other words, and not only the neighbours of the current word. Therefore, incomplete sentences create new dependencies that improve overall prediction accuracy.

As shown also in Table 2, we performed the same experiment with the Random Forest classifier in order to check if the incomplete data instances have a positive effect in the CRF model because of the properties of the algorithm or simply because of the increased dataset size. Our assumption was that since the Random Forest classifier output depends only on local context of a tagged word, it should not be influenced by the new training sequences. This hypothesis was corroborated by our experiment: the classifier trained on the extended dataset performed slightly better, but this difference is much smaller than the one observed for the CRF model with the additional data.

In order to check that the improvements are not only due to the increased dataset size, we performed the same experiments with duplicated training sentences. The output of this duplicated system is identical to the baseline system, showing the key component of the improvement are indeed the incomplete sentences. Our intuition is that since the new training sentences differ from the original ones, they provide new information to the sequence labelling model.

4 Data selection

An inspection of the training and development data showed that 15% of the sentences contain no errors and are thus less useful for model learning. In addition, the majority of the sentences have low edit distance (HTER) score, i.e. contain very few edits/errors. Figure 1 shows the HTER scores distribution for the training dataset: 50% of the sentences have HTER of 0.15 or lower (points below the bottom orange line in the figure), 75% of the sentences have HTER of 0.28 or lower (points below the middle green line). The distributions for the development and test sets are similar.

A large number of sentences with few or no ed-

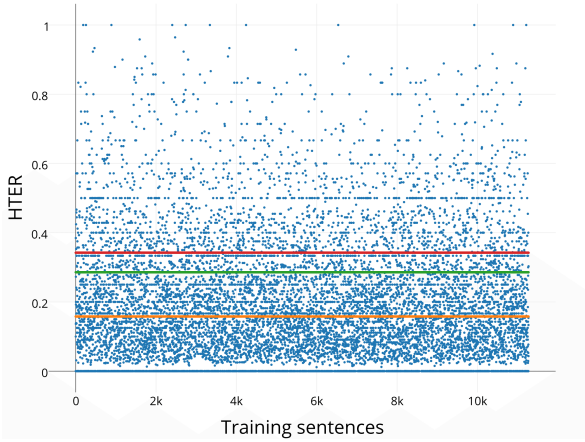

Figure 1: Distribution of HTER scores for the training data: each blue dot represents a training sentence. Dots below the orange line make 50% of the data, dots below the green line, 75% of data, dots above red line, the worst 2000 sentences (18% of the data).

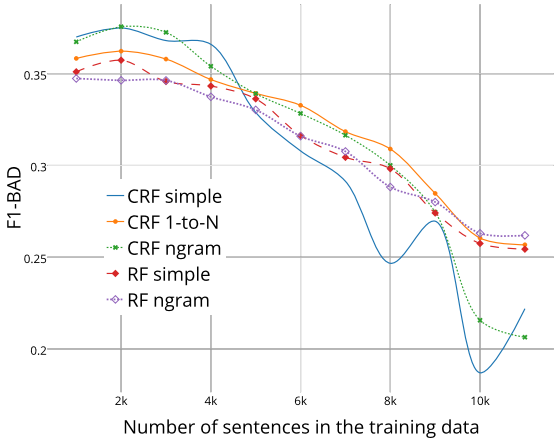

Figure 2: Performance of models trained on subsets of training data (F1 for the "BAD" class).

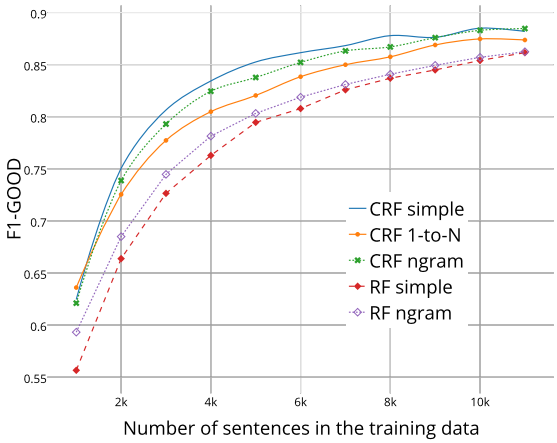

Figure 3: Performance of systems trained on subsets of training data (F1 for the "GOOD" class).

its bias the models to tag more words as "GOOD", i.e. the tagging is too optimistic, which results in higher F1 score for the "GOOD" class and lower F1 score for the "BAD" class. Since our primary goal is improved F1 score for the "BAD" class, we modified the training set to increase the percentage of "BAD" labels.

In order to filter out sentences that have too few errors, we performed a simple training data selection strategy: we used only sentences with the highest amount of editing. To define the optimal number of sentences to select, we built models on different number of training sentences from 1,000 to 11,000 (the entire dataset). Figure 2 shows the learning curves for systems trained on increasing numbers of sentences. Note that the sentences we choose are sorted by their HTER score in decreasing order, i.e. the system trained on 1,000 sentences uses 1,000 sentences with the highest HTER scores (1,000 worst sentences).

Models built trained using only the 2,000 worst sentences have the best F1-BAD score using all learning algorithms. These 2,000 sentences represent 18% of the total available data (data points above the red line in Figure 1). This subset has sentences with HTER scores ranging from 0.34 to 1 and mean value of 0.49.

The highest score is achieved by the CRF model trained on **ngram**-bootstrapped data. However, the data selection strategy changes the effect of bootstrapping that we saw previously: the CRF model without bootstrapping performs very similarly on small data subsets (up to 5,000 sentences), and even outperforms the CRF model with **1-to-N** bootstrapping. On the other hand, a CRF model without bootstrapping is less stable: its quality drops faster as new data is added. The Random Forest classifier has lower prediction accuracy than CRF models, but is more stable than the two models that have the highest scores on 2,000 sentences.

As shown in Figure 3, the learning curves in terms of the F1-score for the "GOOD" class are very different: the scores keeps increasing as we add more training instances. However, after adding 5,000 sentences the growth slows down. Note also that the models that have the least stable F1-BAD scores (CRF without bootstrapping and with **ngram** bootstrapping) show the highest F1-GOOD scores.

5 Official results of shared task

The experiments with data selection (Section 4) showed that all models achieve their highest scores when trained on a subset of 2,000 sentences of the training data with highest HTER. The CRF model with **ngram** bootstrapping yielded the highest F1-BAD of 0.375. We selected this setting as our first submission. Since we could not be sure that the distribution of classes is the same in the development and test sets, for the second submission we chose the same model trained on 5,000 sentences, to reach a balance between the F1-scores for the "BAD" and the "GOOD" classes.

Table 3 summarises the final results. The F1-BAD score of our first system for the test set is 0.366. This submission was ranked 4-th best out of 8. The second system performed worse at tagging the test set: the final F1-BAD score is 0.345, which places it in the 5-th position overall.

		F1-BAD	F1-GOOD	Weigted F1
CRF ngram 2000 sent.	dev	0.375	0.738	0.669
	test	**0.366**	0.744	0.673
CRF ngram 5000 sent.	dev	0.339	0.837	0.742
	test	**0.345**	0.845	0.75

Table 3: Final submission results. Scores in bold were used to compare systems submitted to the shared task.

6 Conclusions

We presented the systems submitted by the DCU-SHEFF team to the word-level QE task at WMT15. Our systems were trained on a set of baseline features released by the organisers of the shared task. We predicted the QE labels using a CRF model trained with CRF++ tool, which was also used to produce the baseline scores.

The main difference between the baseline and our models is that in our systems the training data is filtered prior to training. We use only a small subset of the training sentences which have the highest HTER scores (i.e. the highest percentage of words tagged with the "BAD" label). This led to an increase in the F1 score for the "BAD" class from 0.17 to 0.37.

We also suggested two bootstrapping strategies based on using sub-sequences from the training data as new training instances. These incomplete examples are particularly effective for training CRF models: we were able to improve the F1 score for the "BAD" class from 0.17 to 0.25. However, we were not able to achieve any improvement when the bootstrapping was performed on top of data filtering.

Acknowledgements

This work was supported by the People Programme (Marie Curie Actions) of the European Union's Framework Programme (FP7/2007-2013) under REA grant agreement n^{o} 317471.

References

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit X*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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.

Sylvain Raybaud, David Langlois, and Kamel Smali. 2011. this sentence is wrong. detecting errors in machine-translated sentences. *Machine Translation*, 25(1):1–34.

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