# An Approach Using Style Classification Features for Quality Estimation

Erwan Moreau, Raphaël Rubino

# An approach using style classification features for Quality Estimation

**Erwan Moreau**
CNGL and Computational Linguistics Group
Centre for Computing and Language Studies
School of Computer Science and Statistics
Trinity College Dublin
Dublin 2, Ireland
moreaue@cs.tcd.ie

**Raphael Rubino**
NCLT
Dublin City University
Dublin 9, Ireland
rrubino@computing.dcu.ie

## Abstract

In this paper we describe our participation to the WMT13 Shared Task on Quality Estimation. The main originality of our approach is to include features originally designed to classify text according to some author's style. This implies the use of reference categories, which are meant to represent the quality of the MT output.

## Preamble

*This paper describes the approach followed in the two systems that we submitted to subtask 1.3 of the WMT13 Shared Task on Quality Estimation, identified as* `TCD-DCU-CNGL_1-3_SVM1` *and* `TCD-DCU-CNGL_1-3_SVM2`. *This approach was also used by the first author in his submissions to subtask 1.1, identified as* `TCD-CNGL OPEN` *and* `TCD-CNGL RESTRICTED`[1]. *In the remaining of this paper we focus on subtask 1.3, but there is very little difference in the application of the approach to task 1.1.*

## 1 Introduction

Quality Estimation (QE) aims to provide a quality indicator for machine translated sentences. There are many cases where such an indicator would be useful in a translation process: to compare different Machine Translation (MT) models on a given set of sentences, to tune automatically the parameters of a MT model, to select the bad sentences for human translation or post-editing, to select the good sentences for immediate publication and try to apply automatic post-editing to the others, or simply to provide users who are not fluent in the source language information about the fluency of the translated text they are reading. As long as machine translated text cannot be of reasonably consistent quality, QE is helpful in indicating linguistic quality variability.[2]

After focusing on automatic prediction of ad-hoc quality scores (as estimated by professional annotators) in the previous edition (Callison-Burch et al., 2012), the WMT Shared Task on Quality Estimation 2013 proposes several variants of the task. We participated in task 1.1 which aims to predict HTER scores (edit distance between the MT output and its manually post-edited version), and in task 1.3 which aims to predict the expected time needed to post-edit the MT output.

The originality of our participation lies in the fact that we intended to test "style classification" features for the task of QE: the idea is to select a set of $n$-grams which are particularly representative of a given level of quality. In practice we use only two levels which simply represent low and high quality. We explore various ways to build these two reference categories and to select the $n$-grams, as described in §2. The goal was to see if such features can contribute to the task of predicting quality of MT. As explained in §3, however, various constraints forced us to somehow cut corners in some parts of the features selection and training process; therefore we think that the modest results presented and discussed in §4 might not necessarily reflect the real contribution of these features.

## 2 Features

### 2.1 Classical features

We extract a set of features inspired by the ones provided by the shared task organisers in their 17 baseline feature set. Using the corpora provided for the task, we extract for each source and target

[1] The second author's submission to subtask 1.1 is independent from this approach and is described in a different paper in this volume.

[2] We focus on translation fluency rather than target language faithfulness to sources.



*Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 429–434,
Sofia, Bulgaria, August 8-9, 2013 

segments pair:

- 24 surface features, such as the segment length, the number of punctuation marks and uppercased letters, words with mixed case, etc.
- 30 language Model (LM) features, $n$-gram log-probability and perplexity (with and without start and end of sentence tags) with $n \in [1;5]$.
- 30 backward LM features, $n$-gram log-probability and perplexity (with and without start and end of sentence tags) with $n \in [1;5]$.
- 44 $n$-gram frequency features, with $n \in [1;5]$, extracted from frequency quartiles.
- 24 word-alignment features according to the alignment probability thresholds: 0.01, 0.1, 0.25, 0.5, 0.75 and 1.0, with or without words frequency weighting.

For all these features, except the ones with binary values, we compute the ratio between the source and target feature values and add them to our feature set, which contains 223 classical features.

## 2.2 Style classification features

We call the features described below "style classification" features because they have been used recently in the context of author identification/profiling (Moreau and Vogel, 2013a; Moreau and Vogel, 2013b) (quite sucessfully in some cases). The idea consists in representing the $n$-grams which are very specific to a given "category", a category being a level of quality in the context of QE, and more precisely we use only the *"good"* and *"bad"* categories here.

Thus this approach requires the following parameters:

- At least two datasets used as reference for the categories;
- Various $n$-grams patterns, from which comparisons based on frequency can be done;
- One or several methods to compare a sentence to a category.

### 2.2.1 Reference categories

As reference categories we use both the training datasets provided for task 1.1 and 1.3: both are used in each task, that is, categories are extracted from subtasks 1.1 dataset and 1.3 dataset and used in task 1.1 and 1.3 as well. However we use only half of the sentences of task 1.1 in 1.1 and similarly in 1.3, in order to keep the other half for the classical training process. This is necessary to avoid using (even indirectly) a sentence as both a fixed parameter from which features are extracted (the category data) and an actual instance on which features are computed. In other words this simply follows the principle of keeping the training and test data independent, but in this case there are two stages of training (comparing sentences to a reference category is also a supervised process).

The two datasets are used in three different ways, leading to three distinct pairs of categories "*good/bad*":[3]

- The sentences for which the quality is below the median form the *"bad"* category, the one above form the *"good"* category;
- The sentences for which the quality is below the first quartile form the *"bad"* category, the one above the third quartile form the *"good"* category;
- The complete set of MT output sentences form the *"bad"* category, their manually post-edited counterpart form the *"good"* category.

We use these three different ways to build categories because there is no way to determine a priori the optimal choice. For instance, on the one hand the opposite quartiles probably provide more discriminative power than the medians, but on the other hand the latter contains more data and therefore possibly more useful cases.[4] In the last version the idea is to consider that, in average, the machine translated sentences are of poor quality compared to the manually post-edited sentences; in this case the categories contain more data, but it might be a problem that (1) some of the machine-translated sentences are actually good and (2) the

[3] Below we call "quality" the value given by the HTER score (1.1) or post-editing time (1.3), the level of quality being of course conversely proportional to these values.

[4] The datasets are not very big: only 803 sentences in task 1.3 and 2,254 sentences in task 1.1 (and we can only use half of these for categories, as explained above).

right translation of some difficult phrases in the post-edited sentences might never be found in MT output. We think that the availability of different categories built in various ways is potentially a good thing, because it lets the learning algorithm decide which features (based on a particular category) are useful and which are not, thus tuning the model automatically while possibly using several possibilities together, rather than relying on some predefined categories.

It is important to notice that the correspondence between an MT output and its post-edited version is not used[5]: in all categories the sentences are only considered as an unordered set. For instance it would be possible to use a third-party corpus as well (provided it shares at least a common domain with the data).

We use only the target language (Spanish) of the translation and not the source language in order not to generate too many categories, and because it has been shown that there is a high correlation between the complexity of the source sentence and the fluency of the translation (Moreau and Vogel, 2012). However it is possible to do so for the categories based on quantiles.

#### 2.2.2 $n$-grams patterns, thresholds and distance measures

We use a large set of 30 $n$-grams patterns based on tokens and POS tags. POS tagging has been performed with TreeTagger (Schmid, 1995). Various combinations of $n$-grams are considered, including standard sequential $n$-grams, skip-grams, and combinations of tokens and POS tags.

Since the goal is to compare a sentence to a category, we consider the frequency in terms of number of sentences in which the $n$-gram appears, rather than the global frequency or the local frequency by sentence.[6]

Different frequency thresholds are considered, from 1 to 25. Additionally we can also filter out $n$-grams for which the relative frequency is too similar between the *"good"* and *"bad"* categories. For instance it is possible to keep only the $n$-grams for which 80% of the occurrencies belong to the *"bad"* category, thus making it a strong marker for low quality. Once again different thresholds are considered, in order to tradeoff between the amount of cases and their discriminative power.

We use only three simple distance/similarity measures when comparing a sentence to a category:

- Binary match: for each $n$-gram in the sentence, count 1 if it belongs to the category, 0 otherwise, then divide by the number of $n$-grams in the sentence;
- Weighted match: same as above but sum the proportion of occurrences belonging to the category instead of 1 (this way an $n$-gram which is more discriminative is given more weight);
- Cosine similarity.

Finally for every tuple formed by the combination of

- a category,
- a quality level (*"good/bad"*),
- an $n$-gram pattern,
- a frequency threshold,
- a threshold for the proportion of the occurrences in the given category,
- and a distance measure

a feature is created. For every sentence the value of the feature is the score computed using the parameters defined in the tuple. From our set of parameters we obtain approximately 35,000 features.[7] It is worth noticing that these features are not meant to represent the sentence entirely, but rather particularly noticeable parts (in terms of quality) of the sentence.

[5] in the categories used as reference data; but it is used in the final features during the (supervised) training stage (see §3).

[6] The frequency by sentence is actually also taken into account in the following way: instead of considering only the $n$-gram, we consider a pair ($n$-gram, local frequency) as an observation. This way if a particular frequency is observed more often in a given category, it can be interpreted as a clue in favor of this category. However in most cases (long $n$-grams sequences) the frequency by sentence is almost always one, sometimes two. Thus this is only marginally a relevant criterion to categorize a sentence.

[7] The number of features depends on the data in the category, because if no $n$-gram at all in the category satisfies the conditions given by the parameters (which can happen with very high thresholds), then the feature does not exist.

### 2.3 Features specific to the dataset

In task 1.3 we are provided with a translator id and a document id for each sentence. The distribution of the time spent to post-edit the sentence depending on these parameters shows some significant differences among translators and documents. This is why we add several features intended to account for these parameters: the id itself, the mean and the median for both the translator and the document.

## 3 Design and training process

The main difficulty with so many features (around 35,000) is of course to select a subset of reasonable size, in order to train a model which is not overfitted. This requires an efficient optimization method, since it is clearly impossible to explore the search space exhaustively in this case.

Initially it was planned to use an ad-hoc genetic algorithm to select an optimal subset of features. But unfortunately the system designed in this goal did not work as well as expected[8], this is why we had to switch to a different strategy: the two final sets of features were obtained through several stages of selection, mixing several different kinds of correlation-based features selection methods.

The different steps described below were carried out using the Weka Machine Learning toolkit[9] (Hall et al., 2009). Since we have used half of the training data as a reference corpus for some of the categories (see §2), we use the other half as training instances in the selection and learning process, with 10 folds cross-validation for the latter.

### 3.1 Iterative selection of features

Because of the failure of the initial strategy, in order to meet the time constraints of the Shared Task we had to favor speed over performance in the process of selecting features and training a model. This probably had a negative impact on the final results, as discussed in section §4.

In particular the amount of features was too big to be processed in the remaining time by a subset selection method. This is why the features were first ranked individually using the Relief attribute estimation method (Robnik-Sikonja and Kononenko, 1997). Only the 20,000[10] top features were extracted from this ranking and used further in the selection process.

From this initial subset of features, the following heuristic search algorithms combined with a correlation-based method[11] to evaluate subsets of features (Hall, 1998) are applied iteratively to a given input set of features:

- Best-first search (forward, backward, bidirectional);
- Hill-climbing search (forward and backward);
- Genetic search with Bayes Networks.

Each of these algorithms was used with different predefined parameters in order to trade off between time and performance. This selection process is iterated as long as the number of features left is (approximately) higher than 200.

### 3.2 Training the models

When less than 200 features are obtained, the iterative selection process is still applied but a 10 folds cross-validated evaluation is also performed with the following regression algorithms:

- Support Vector Machines (SVM) (Smola and Schölkopf, 2004; Shevade et al., 2000);
- Decision trees (Quinlan, 1992; Wang and Witten, 1996);
- Pace regression (Wang and Witten, 2002).

These learning algorithms are also run with several possible sets of parameters. Eventually the submitted models are chosen among those for which the set of features can not be reduced anymore without decreasing seriously the performance. Most of the best models were obtained with SVM, although the decision trees regression algorithm performed almost as well. It was not possible to decrease the number of features below 60 for task 1.3 (80 for task 1.1) without causing a loss in performance.

[8] At the time of writing it is still unclear if this was due to a design flaw or a bug in the implementation.

[9] Weka 3.6.9, `http://www.cs.waikato.ac.nz/ml/weka`.

[10] For subtask 1.3. Only the 8,000 top features for subtask 1.1.

[11] Weka class `weka.attributeSelection.CfsSubsetEval`.

## 4 Results and discussion

The systems are evaluated based on the Mean Average Error, and every team was allowed to submit two systems. Our systems ranked 10th and 11th among 14 for task 1.1, and 13th and 15th among 17 for task 1.1.

### 4.1 Possible causes of loss in performance

We plan to investigate why our approach does not perform as well as others, and in particular to study more exhaustively the different possibilities in the features selection process.[12] It is indeed very probable that the method can perform better with an appropriate selection of features and optimization of the parameters, in particular:

- The final number of features is too large, which can cause overfitting. Most QE system do not need so many features (only 15 for the best system in the WMT12 Shared Task on QE (Soricut et al., 2012)).
- We had to perform a first selection to discard some of the initial features based on their individual contribution. This is likely to be a flaw, since some features can be very useful in conjuction with other even if poorly informative by themselves.
- We also probably made a mistake in applying the selection process to the whole set of features, including both classical features and style classification features: it might be relevant to run two independent selection processes at first and then gather the resulting features together only for a more fine-grained final selection. Indeed, the final models that we submitted include very few classical features; we believe that this might have made these models less reliable, since our initial assumption was rather that the style classification features would act as secondary clues in a model primarily relying on the classical features.

### 4.2 Selected features

The following observations can be made on the final models obtained for task 1.3, keeping in mind that the models might not be optimal for the reasons explained above:

- Only 5% of the selected features are classical features;
- The amount of data used in the category seems to play an important role: most features correspond to categories built from the 1.1 dataset (which is bigger), and the proportions between the different kinds of categories are: 13% for first quartile vs. fourth quartile (smallest dataset), 25% for below median vs. above median, and 61% for MT output vs. postedited sentence (largest dataset);
- It seems more interesting to identify the low quality $n$-grams (i.e. errors) rather than the high quality ones: 76% of the selected features represent the *"bad"* category;
- 81% of the selected features represent an $n$-grams containing at least one POS tag, whereas only 40% contain a token;
- Most features correspond to selecting $n$-grams which are very predictive of the *"good/bad"* category (high difference of the relative proportion between the two categories), although a significant number of less predictive $n$-grams are also selected;
- The cosine distance is selected about three times more often than the two other distance methods.

## 5 Conclusion and future work

In conclusion, the approach performed decently on the Shared Task test data, but was outperformed by most other participants systems. Thus currently it is not proved that style classification features help assessing the quality of MT. However the approach, and especially the contribution of these features, have yet to be evaluated in a less constrained environment in order to give a well-argued answer to this question.

## Acknowledgments

This research is supported by the Science Foundation Ireland (Grant 12/CE/I2267) as part of the Centre for Next Generation Localisation (`www.cngl.ie`) funding at Trinity College, University of Dublin.

[12] Unfortunately the results of this study are not ready yet at the time of writing.

## References


Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, Montreal, Canada, June. Association for Computational Linguistics.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. 2009. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

M. A. Hall. 1998. *Correlation-based Feature Subset Selection for Machine Learning*. Ph.D. thesis, University of Waikato, Hamilton, New Zealand.

Erwan Moreau and Carl Vogel. 2012. Quality estimation: an experimental study using unsupervised similarity measures. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 120–126, Montréal, Canada, June. Association for Computational Linguistics.

Erwan Moreau and Carl Vogel. 2013a. Participation to the pan author identification task. In *to appear in the proceeding of CLEF 2013*.

Erwan Moreau and Carl Vogel. 2013b. Participation to the pan author profiling task. In *to appear in the proceeding of CLEF 2013*.

J.R. Quinlan. 1992. Learning with continuous classes. In *Proceedings of the 5th Australian joint Conference on Artificial Intelligence*, pages 343–348. Singapore.

Marko Robnik-Sikonja and Igor Kononenko. 1997. An adaptation of relief for attribute estimation in regression. In Douglas H. Fisher, editor, *Fourteenth International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

S.K. Shevade, SS Keerthi, C. Bhattacharyya, and K.R.K. Murthy. 2000. Improvements to the SMO algorithm for SVM regression. *Neural Networks, IEEE Transactions on*, 11(5):1188–1193.

A.J. Smola and B. Schölkopf. 2004. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Radu Soricut, Nguyen Bach, and Ziyuan Wang. 2012. The SDL Language Weaver systems in the WMT12 Quality Estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 145–151, Montréal, Canada, June. Association for Computational Linguistics.

Y. Wang and I.H. Witten. 1996. Induction of model trees for predicting continuous classes.

Y. Wang and I.H. Witten. 2002. Modeling for optimal probability prediction. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 650–657. Morgan Kaufmann Publishers Inc.