LORIA System for the WMT13 Quality Estimation Shared Task

Langlois David
LORIA
(Université de Lorraine, INRIA, CNRS)
615 rue du Jardin Botanique,
54602 Villers les Nancy, France
david.langlois@loria.fr

Smaïli Kamel
LORIA
(Université de Lorraine, INRIA, CNRS)
615 rue du Jardin Botanique,
54602 Villers les Nancy, France
kamel.smaili@loria.fr

Abstract

In this paper we present the system we submitted to the WMT13 shared task on Quality Estimation. We participated in the Task 1.1. Each translated sentence is given a score between 0 and 1. The score is obtained by using several numerical or boolean features calculated according to the source and target sentences. We perform a linear regression of the feature space against scores in the range [0..1]. To this end, we use a Support Vector Machine with 66 features. In this paper, we propose to increase the size of the training corpus. For that, we use the post-edited and reference corpora during the training step. We assign a score to each sentence of these corpora. Then, we tune these scores on a development corpus. This leads to an improvement of 10.5% on the development corpus, in terms of Mean Average Error, but achieves only a slight improvement on the test corpus.

1 Introduction

In the scope of Machine Translation (MT), Quality Estimation (QE) is the task consisting to evaluate the translation quality of a sentence or a document. This process may be useful for post-editors to decide or not to revise a sentence produced by a MT system (Specia, 2011; Specia et al., 2010). Moreover, it can be useful to decide if a translated document can be broadcasted or not (Soricut and Echihabi, 2010). The most obvious way to give a score to a translated sentence consists in using a machine learning approach. This approach is supervised: experts are asked to score translated sentences and with the obtained material, one learns a prediction model of scores. The main drawback of the machine learning approach is that it is supervised and requires huge data. To score a sentence is time-consuming. Moreau et al. in (Moreau and Vogel, 2012) dealt with this issue by proposing unsupervised similarity measures. In fact, the score of a translated sentence is defined by a measure giving the distance between it and the contents of an external corpus. The authors improve the results of the supervised approach but this method can be used only in the ranking task. Raybaud et al. (Raybaud et al., 2011) proposed a method to add errors in reference sentences (deletion, substitution, insertion). By this way, they build additional corpus in which each word can be associated with a label correct/not correct. But, it is not possible to predict the translation quality of sentences including these erroneous words.

In this paper, we propose to increase the size of the training corpus. For that, we use the score given by experts to evaluate additional sentences from the post-edited and reference corpora. Practically, we extract from source and target sentences numerical vectors (features) and we learn a prediction model of the scores. Then, we apply this model to predict the scores of the post-edited and the reference sentences. And finally, we tune the predicted scores on a development corpus.

The article is structured as follows. In Section 2, we give an overview of our machine learning approach and of the features we use. Then, in Sections 3 and 4 we describe the corpora and how we increase the size of the training corpus by a partly-unsupervised approach. In section 5, we give results about this method and we end by a conclusion and perspectives.

2 Overview of our quality estimation submission

We submit a system for the task 1.1: one has to evaluate each translated sentence with a score between 0 and 1. This score is read as the HTER between the translated sentence and its post-edited version. Each translated sentence is assigned a

Proceedings of the Eighth Workshop on Statistical Machine Translation, pages 380–385,
Sofia, Bulgaria, August 8-9, 2013

score between 0 and 1. The score is calculated using several numerical or boolean features extracted according to the source and target sentences. We perform a regression of the feature space against $[0..1]$. To this end, we use the Support Vector Machine algorithm (LibSVM toolkit (Chang and Lin, 2011)). We experimented only the linear kernel because our experience from last year (Langlois et al., 2012) showed that its performance are yet good while no parameters have to be tuned on a development corpus.

2.1 The baseline features

The QE shared task organizers provided a baseline system including the same features as last year: source and target sentences lengths; average source word length; source and target likelihood computed with 3-gram (source) and 5-gram (target) language models; average number of occurrences of the words within the target sentence; average number of translations per source word in the sentence, using IBM1 translation table (only translations higher than 0.2); weighted average number of translations per source word in the sentence (similar to the previous one, but a frequent word is given a low weight in the averaging); distribution by frequencies of the source n-gram into the quartiles; match between punctuation in source and target. Overall, the baseline system proposes 17 features. We remark that only 5 features take into account the target sentence.

2.2 The LORIA features

In previous works (Raybaud et al., 2011; Langlois et al., 2012), we tested several confidence measures. As last year (Langlois et al., 2012), we use the same features. We extract information by the way of language model (perplexity, level of back-off, intra-lingual triggers) and translation table (IBM1 table, inter-lingual triggers). The features are defined at word level, and the features at sentence level are computed by averaging over each word in the sentence. In our system, we use, in addition to baseline features, ratio of source and target lengths; source and target likelihood computed with 5-gram language models (Duchateau et al., 2002) (in addition to 3-gram features from baseline); level of backoff *n*-gram based features (Uhrik and Ward, 1997). This feature indicates if the 3-gram, the 2-gram or the unigram corresponding to the word is in the language model. For likelihoods and levels of backoff, we use models trained on corpus read from left to right (classical way), and from right to left (sentences are reversed before training language models). This leads to two language models, and therefore to two values for each feature and side (source and target). Moreover, a common property of all *n*-gram and backoff based features is that a word can get a low score if it is actually correct but its neighbours are wrong. To compensate for this phenomenon we took into account the average score of the neighbours of the word being considered. More precisely, for every relevant feature $x_.$ defined at word level we also computed:

$$\begin{aligned} x_.^{left}(w_i) &= x_.(w_{i-2}) * x_.(w_{i-1}) * x_.(w_i) \\ x_.^{centred}(w_i) &= x_.(w_{i-1}) * x_.(w_i) * x_.(w_{i+1}) \\ x_.^{right}(w_i) &= x_.(w_{i)} * x_.(w_{i+1}) * x_.(w_{i+2}) \end{aligned}$$

The other features are intra-lingual features: each word is assigned its average mutual information with the other words in the sentence; inter-lingual features: each word in target sentence is assigned its average mutual information with the words in source sentence; IBM1 features: contrary to IBM1 based baseline features which take into account the number of translations, we use the probability values in the translation table between source and target words; basic parser (correction of bracketing, presence of end-of-sentence symbol); number and ratio of out-of-vocabulary words in source and target sentences. This leads to 49 features. A few ones are equivalent to or are strongly correlated to baseline ones. We remark that 27 features take into account the target sentence.

The union of the both sets baseline+loria improved slightly the baseline system on the test set provided by the QE Shared Task 2012 (Callison-Burch et al., 2012).

3 Corpora

The organizers provide a set of files for training and development. We list below the ones we used:

- source.eng: 2,254 source sentences taken from three WMT data sets (English): news-test2009, news-test2010, and news-test2012. In the following, this file is named `src`
- target_system.spa: translations for the source sentences (Spanish) generated by a PB-SMT system built using Moses. In the following, this file is named `syst`

- target_system.HTER_official-score: HTER scores between MT and post-edited version, to be used as the official score in the shared task. In the following, this file is named `hteroff`
- target_reference.spa: reference translation (Spanish) for source sentences as originally given by WMT; In the following, this file is named `ref`
- target_postedited.spa: human post-edited version (Spanish) of the machine translations in target_system.spa. In the following, this file is named `post`

We split these files into two parts: a training part made up of the 1,832 first sentences, and a development part made up of the 442 remaining sentences. This choice is motivated by the fact that in the previous evaluation campaign we had exactly the same experimental conditions.
For each given file `f`, we use therefore a part named `f.train` for training and a part named `f.dev` for development.

4 Training Algorithm

This section describes the approach we propose to increase the size of the training corpus.

We have to train the prediction model of scores from the source and target sentences.

The common way to train such a prediction model consists in extracting a features vector for each couple `(source,target)` from the `(src.train,syst.train)` corpus. For each vector, the score associated by experts to the corresponding sentence is assigned. Then, we use a machine learning approach to learn the regression between the vectors and the scores. And finally, we use the triplet `(src.dev,syst.dev,hteroff.dev)` to tune parameters.

With machine learning approach, the number of examples is crucial for a relevant training, but unfortunately the evaluation campaign provides a training corpus of only 1,832 examples.

To increase the training corpus, we propose to use the `ref` and `post` files. But for that, we have to associate a score to these new target sentences. One way could be to calculate the HTER score between each sentence and its corresponding sentence in the post edited file. But this leads to a drawback: all the couples `(src,post)` would have a score equal to 0, and then there is a risk of overtraining on the 0 value. To prevent this problem, we preferred to learn a prediction model from the `(src.train,syst.train,hteroff.train)` triplet. Then we apply this prediction model to the `(src.train,post.train)` and to the `(src.train,ref.train)`. By this way, we get a training corpus made up of $1,832 \times 3 = 3,696$ examples with their scores. Consequently, it is possible to learn a prediction model from this new training corpus. These scores are not optimal because the features cannot describe all the information from sentences, and a machine learning approach is limited if data are not sufficiently huge. Therefore, we propose an anytime randomized algorithm to tune the reference and post-edited scores on the development corpus. We give below the algorithm we propose.

1. **Prediction model**
   (a) Learn the prediction model using only features from `(src.train,syst.train)` and HTER target scores from experts
2. **Predict initial scores for postedited and reference sentences**
   (a) Use this model to predict the scores associated to the features from `(src.train,post.train)` and `(src.train,ref.train)`. The predicted scores for `(src.train,post.train)` are called `post_best` and the ones for `(src.train,ref.train)` are called `ref_best`
3. **Learn initial prediction model using the 3 trains (system part, post-edited part and reference part)**
   (a) Learn the prediction model using features from the three sets of features and the scores associated to these sets (experts scores, `post_best` and `ref_best`)
   (b) Evaluate this model. This leads to a performance equal to `best`
4. **Tune scores for postedited and reference sentences**
   (a) Repeat the following steps until stop

(b) Build a new set of scores named `post_new` (resp. `ref_new`) by disturbing each score of `post_best` (resp. `ref_best`) with a probability equal to `pdisturb`. A modified score is shifted by a value randomly chosen in `[-disturb,+disturb]`

(c) Learn the prediction model using features from the three sets of features and the new scores associated to these sets (experts scores for system set, `post_new` and `ref_new` for the post-edited and reference sets)

(d) Evaluate this model. This leads to a performance equal to `perf`

(e) If `perf<best` then replace `best` by `perf`, `post_best` by `post_new` and `ref_best` by `ref_new`.

To evaluate a model, we use it to predict the scores on the development corpus. Then we compare the predicted scores to the expert scores and we compute the Mean Average Error (MAE) given by the formula $MAE(s,r) = \frac{\sum_{i=1}^{n} |s_i - r_i|}{n} \times 100$ where s and r are two sets of n scores.

5 Results

We used the data provided by the shared task on QE, without additional corpus. This data is composed of a parallel English-Spanish training corpus. This corpus is made of the concatenation of europarl-v5 and news-commentary10 corpora (from WMT-2010), followed by tokenization, cleaning (sentences with more than 80 tokens removed) and truecasing. It has been used for baseline models provided in the baseline package by the shared task organizers. We used the same training corpus to train additional language models (5-gram with kneyser-ney discounting, obtained with the SRILM toolkit) and triggers required for our features. For feature extraction, we used the files provided by the organizers: 2,254 source english sentences, their translations by the baseline system, and the score of these translations. This score is the HTER between the proposed translation and the post-edited sentence. We used the train part to perform the regression between the features and the scores. Therefore, the system we propose in this campaign is the same as the one we presented for the previous campaign in terms of features. But, we only use a SVM with a linear kernel and we do not use any feature selection. The added value of the new system is the fact that we increase the size of the training corpus.

To evaluate the different configurations, we used the MAE measure. The performance of our system with only the classical train set `(src.train,syst.train)` are given in Table 1. In this table, BASELINE+LORIA use both features BASELINE and LORIA (Section 2). We remark that, contrary to last year, the BASELINE+LORIA do not improve the performance of the BASELINE features on the development set.

Set of features	**Dev**
BASELINE	13.46
LORIA	14.04
BASELINE+LORIA	13.88

Table 1: Performance in terms of MAE without increasing the training corpus

Now, we increase the training corpus with the method described in previous section. First, we use the system trained on `(src.train,syst.train)` to predict scores for the sentences in `post.train` and `ref.train`. We know that these scores should represent the HTER score, then a well translated sentence should be assigned a higher score. Therefore, we can make the hypothesis that sentences from `post.train` and `ref.train` are better than those in `syst.train`. We check this hypothesis by comparing the distributions of HTER scores in the three files (true HTER scores in `syst.train`, and predicted scores in the two other files). We present in Table 2 the Minimum, Maximum, Mean and Standard Deviation of this score for the three corpora. We remark that the scores are not well predicted because some of them are negative while all scores in `syst.train` are between 0 and 1. This is due to the fact that the constraint of HTER in terms of limit values is not explicitly taken into account by SVM. We give more details about these scores out of $[0..1]$ in Table 3. For `post.train`, 2 scores are under 0 with a mean value equal to -0.123, and no scores are higher than 1. For `ref.train`, 4 scores are under 0 with a mean value equal to -3.023, and 26 scores are higher than 1 with a mean equal to 1.126. Comparing to the 1,832 sentences in the training corpus, we can conclude that the 'outliers' are very rare. In Table 2 Mean

and Standard Deviation are computed only for scores predicted between 0 and 1. The obtained mean values are quite similar, but the standard deviation is very low for predicted scores.

This configuration leads to a performance equal to 13.88 on the development corpus, which is slightly worse than the BASELINE system but slightly better than the BASELINE+LORIA system.

Because, SVM predicts scores which do not represent exactly HTER and because the model is learnt on a relatively small corpus (1,832 sentences), we decided to modify randomly some scores. This operation is called in the following the tuning process.

Set	Min	Max	Mean	SD
syst. train	0	1	0.317	0.169
post. train	-0.147	0.708	0.315	0.083
ref. train	-11.314	0.746	0.329	0.081

Table 2: Statistics on HTER for the three sets of sentences used in the training corpus

	lower than 0		higher than 1	
Set	**Nb**	**Mean**	**Nb**	**Mean**
syst.train	0	-	0	-
post.train	2	-0.123	0	-
ref.train	4	-3.023	26	1.126

Table 3: Statistics on HTER for the three sets of sentences used in the training corpus. Nb is the number of sentences

For the tuning process, after several tests, we fixed to 0.1 the probability `pdisturb` to modify the score of a sentence. Then, the score is modified by randomly shifting it in $[-0.01... + 0.01]$. We start with the initial predicted scores (MAE = 13.88). Then we randomly modify a subset of scores and keep a new configuration if its MAE is improved. The process is stopped when MAE converges. Figure 1 presents the evolution of MAE on the development corpus.

The process stopped after $22,248$ iterations. Only 274 (1.2%) iterations led to an improvement. We present the results of this approach on the development corpus and on the official test set of the

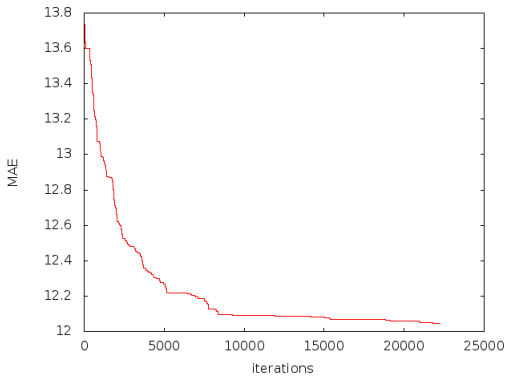

Figure 1: Evolution of the MAE on the development corpus

campaign (500 sentences). We group in Table 4 the results on development and test corpus for the BASELINE features and the BASELINE+LORIA features with and without using the post-edited and reference sentences. Finally, we achieve a MAE of 12.05 on the development set. This constitutes an improvement of 10.5% in comparison to the BASELINE system. But we improve only slightly the performance of the baseline system on the test set. We conclude that there is an overtraining on the development corpus. In order to prevent from this problem, we could use a leaving-one-out approach on training and development corpora.

With the tuned values of scores, we calculated the same statistics as in Tables 2 and 3. We present these statistics in Tables 5 and 6. As we can see, the tuning process leads to an increasing of the mean value of the scores. Moreover, the number of scores out of range increases. This analysis reinforces our conclusion about overtraining: predicted scores may be strongly modified to obtain a good performance on the development corpus.

Set of features	Dev	Test
BASELINE	13.46	14.81
BASELINE+LORIA	13.88	nc
+ postedited + ref	13.78	nc
+ tuning	12.05	14.79

Table 4: Performance in terms of MAE of the features with and without increasing the training corpus

To conclude the experiments, we try to fix the problem of scores predicted out of range. For that, we set to 0 the scores lower than 0 and to 1 the

Set	Min	Max	Mean	SD
post. train	-0.811	1.322	0.407	0.235
ref. train	-10.485	1.320	0.409	0.242

Table 5: Statistics on HTER for the post and ref sets of sentences used in the training corpus, after tuning

	lower than 0		higher than 1	
Set of sentences	**Nb**	**Mean**	**Nb**	**Mean**
post.train	318	-0.164	29	1.118
ref.train	282	-0.205	28	1.123

Table 6: Statistics on HTER for the post and ref sets of sentences used in the training corpus, after tuning. Nb is the number of sentences.

ones greater than 1. Then we learn a new SVM model using these new scores. This leads to a MAE equal to 12.18 on the development corpus and 14.83 on the test corpus, which is worse than the performance without correction. This is for us a drawback of the machine learning approach. For this approach, the scores have no semantic. SVM do not "know" that the scores are HTER between 0 and 1. Then, if tuning leads to no reasonable values, this is not a problem if it increases the performance. Moreover, maybe the features do not extract from all sentences information representative of their quality, and this quality is overestimated: then the tuning system has to lower strongly the corresponding scores to counteract this problem.

6 Conclusion and perpespectives

In this paper we propose a method to increase the size of the training corpus for QE in the scope of Task 1.1. We add to the initial training corpus (sentences translated by a machine translation system) the post-edited and the reference sentences. We associate to these sentences scores predicted by using a model learnt on the system sentences. Then we tune the predicted scores on the development corpus. This method leads to an improvement of 10.5% on the development corpus in terms of MAE, but achieves only a slight improvement on the test corpus. A statistical study shows that tuning scores leads to out of range values. This surprising behavior have to be investigated. In addition, we will test another machine learning tools (neural networks for example). Another point is that, contrary to last year, the whole set of features leads to worse performance than baseline features. This could be explained by the fact that no selecting algorithm has been used to choose the best features. In fact, we preferred, this year to investigate the underlying knowledge on the post-edited and reference corpora. Last, we conclude that the good improvement on the development corpus is not reproduced on the test corpus. In order to prevent from this problem, we will use a leaving-one-out approach on the training.

References

C. Callison-Burch, P. Koehn, C. Monz, M. Post, R. Soricut, and L. Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51.

C.-C. Chang and C.-J. Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

J. Duchateau, K. Demuynck, and P. Wambacq. 2002. Confidence scoring based on backward language models. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 221–224.

D. Langlois, S. Raybaud, and Kamel Smaïli. 2012. Loria system for the WMT12 quality estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 114–119.

E. Moreau and C. Vogel. 2012. Quality estimation: an experimental study using unsupervised similarity measures. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 120–126.

S. Raybaud, D. Langlois, and K. Smaïli. 2011. "This sentence is wrong." Detecting errors in machine-translated sentences. *Machine Translation*, 25(1):1–34.

R. Soricut and A. Echihabi. 2010. Trustrank: Inducing trust in automatic translations via ranking. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 612–621.

L. Specia, N. Hajlaoui, C. Hallett, and W. Aziz. 2010. Predicting machine translation adequacy. In *Proceedings of the Machine Translation Summit XIII*, pages 612–621.

L. 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.

C. Uhrik and W. Ward. 1997. Confidence metrics based on n-gram language model backoff behaviors. In *Fifth European Conference on Speech Communication and Technology*, pages 2771–2774.