LIG System for WMT13 QE Task: Investigating the Usefulness of Features in Word Confidence Estimation for MT

Ngoc-Quang Luong **Benjamin Lecouteux** **Laurent Besacier**
LIG, Campus de Grenoble
41, Rue des Mathématiques,
UJF - BP53, F-38041 Grenoble Cedex 9, France
{ngoc-quang.luong,laurent.besacier,benjamin.lecouteux}@imag.fr

Abstract

This paper presents the LIG's systems submitted for Task 2 of WMT13 Quality Estimation campaign. This is a word confidence estimation (WCE) task where each participant was asked to label each word in a translated text as a binary (Keep/Change) or multi-class (Keep/Substitute/Delete) category. We integrate a number of features of various types (system-based, lexical, syntactic and semantic) into the conventional feature set, for our baseline classifier training. After the experiments with all features, we deploy a "Feature Selection" strategy to keep only the best performing ones. Then, a method that combines multiple "weak" classifiers to build a strong "composite" classifier by taking advantage of their complementarity is presented and experimented. We then select the best systems for submission and present the official results obtained.

1 Introduction

Recently Statistical Machine Translation (SMT) systems have shown impressive gains with many fruitful results. While the outputs are more acceptable, the end users still face the need to post edit (or not) an automatic translation. Then, the issue is to be able to accurately identify the correct parts as well as detecting translation errors. If we focus on errors at the word level, the issue is called Word-level Confidence Estimation (WCE).

In WMT 2013, a shared task about quality estimation is proposed. This quality estimation task is proposed at two levels: word-level and sentence-level. Our work focuses on the word-level quality estimation (named Task 2). The objective is to highlight words needing post-edition and to detect parts of the sentence that are not reliable. For the task 2, participants produce for each token a label according to two sub-tasks:

- a binary classification: good (keep) or bad (change) label
- a multi-class classification: the label refers to the edit action needed for the token (i.e. keep, delete or substitute).

Various approaches have been proposed for WCE: Blatz et al. (2003) combine several features using neural network and naive Bayes learning algorithms. One of the most effective feature combinations is the Word Posterior Probability (WPP) as proposed by Ueffing et al. (2003) associated with IBM-model based features (Blatz et al., 2004). Ueffing and Ney (2005) propose an approach for phrase-based translation models: a phrase is a sequence of contiguous words and is extracted from word-aligned bilingual training corpus. The confidence value of each word is then computed by summing over all phrase pairs in which the target part contains this word. Xiong et al. (2010) integrate target word's Part-Of-Speech (POS) and train them by Maximum Entropy Model, allowing significative gains compared to WPP features. Other approaches are based on external features (Soricut and Echihabi, 2010; Felice and Specia, 2012) allowing to deal with various MT systems (e.g. statistical, rule based etc.).

In this paper, we propose to use both internal and external features into a conditionnal random fields (CRF) model to predict the label for each word in the MT hypothesis. We organize the article as follows: section 2 explains all the used features. Section 3 presents our experimental settings and the preliminary experiments. Section 4 explores a feature selection refinement and the section 5 presents work using several classifiers associated with a boosting decision. Finally we present our systems submissions and propose some conclusions and perspectives.

2 Features

In this section, we list all 25 types of features for building our classifier (see a list in Table 3). Some of them are already used and described in detail in our previous paper (Luong, 2012), where we deal with French - English SMT Quality Estimation. WMT13 was a good chance to re-investigate their usefulness for another language pair: English-Spanish, as well as to compare their contributions with those from other teams. We categorize them into two types: the **conventional features**, which are proven to work efficiently in numerous CE works and are inherited in our systems, and the **LIG features** which are more specifically suggested by us.

2.1 The conventional features

We describe below the conventional features we used. They can be found in some previous papers dealing with WCE.

- Target word features: the target word itself; the bigram (trigram) it forms with one (two) previous and one (two) following word(s); its number of occurrences in the sentence.
- Source word features: all the source words that align to the target one, represented in BIO[1] format.
- Source alignment context features: the combinations of the target word and one word before (left source context) or after (right source context) the source word aligned to it.
- Target alignment context features: the combinations of the source word and each word in the window ± 2 (two before, two after) of the target word.
- Target Word's Posterior Probability (WPP).
- Backoff behaviour: a score assigned to the word according to how many times the target Language Model has to back-off in order to assign a probability to the word sequence, as described in (Raybaud et al., 2011).
- Part-Of-Speech (POS) features (using TreeTagger[2] toolkit): The target word's POS; the source POS (POS of all source words aligned to it); bigram and trigram sequences between its POS and the POS of previous and following words.
- Binary lexical features that indicate whether the word is a: *stop word* (based on the stop word list for target language), *punctuation* symbol, *proper name* or *numerical*.

2.2 The LIG features

- Graph topology features: based on the N-best list graph merged into a confusion network. On this network, each word in the hypothesis is labelled with its WPP, and belongs to one *confusion set*. Every completed path passing through all nodes in the network represents one sentence in the N-best, and must contain exactly one link from each confusion set. Looking into a confusion set, we find some useful indicators, including: the *number of alternative paths* it contains (called *Nodes*), and the distribution of posterior probabilities tracked over all its words (most interesting are *maximum and minimum probabilities*, called *Max* and *Min*).
- Language Model (LM) features: the *"longest target n-gram length"* and *"longest source n-gram length"*(length of the longest sequence created by the current target (source aligned) word and its previous ones in the target (source) LM). For example, with the target word w_i: if the sequence $w_{i-2}w_{i-1}w_i$ appears in the target LM but the sequence $w_{i-3}w_{i-2}w_{i-1}w_i$ does not, the n-gram value for w_i will be 3.
- The *word's constituent label* and *its depth in the tree* (or the distance between it and the tree root) obtained from the constituent tree as an output of the Berkeley parser (Petrov and Klein, 2007) (trained over a Spanish treebank: AnCora[3]).
- Occurrence in Google Translate hypothesis: we check whether this target word appears in the sentence generated by Google Translate engine for the same source.

[1]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/

[2]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

[3]http://clic.ub.edu/corpus/en/ancora

- Polysemy Count: the *number of senses* of each word given its POS can be a reliable indicator for judging if it is the translation of a particular source word. Here, we investigate the polysemy characteristic in both target word and its aligned source word. For source word (English), the number of senses can be counted by applying a Perl extension named Lingua::WordNet[4], which provides functions for manipulating the WordNet database. For target word (Spanish), we employ BabelNet[5] - a multilingual semantic network that works similarly to WordNet but covers more European languages, including Spanish.

3 Experimental Setting and Preliminary Experiment

The WMT13 organizers provide two bilingual data sets, from English to Spanish: the training and the test ones. The training set consists of 803 MT outputs, in which each token is annotated with one appropriate label. In the binary variant, the words are classified into *"K"* (Keep) or *"C"* (Change) label, meanwhile in the multi-class variant, they can belong to *"K"* (Keep), *"S"* (Substitution) or *"D"* (Deletion). The test set contains 284 sentences where all the labels accompanying words are hidden. For optimizing parameters of the classifier, we extract 50 sentences from the training set to form a development set. Since a number of repetitive sentences are observed in the original training set, the dev set was carefully chosen to ensure that there is no overlap with the new training set (753 sentences), keeping the tuning process accurate. Some statistics about each set can be found in Table 1.

Motivated by the idea of addressing WCE as a sequence labeling task, we employ the *Conditional Random Fields* (CRF) model (Lafferty et al., 2001) and the corresponding WAPITI toolkit (Lavergne et al., 2010) to train our classifier. First, we experiment with the combination of all features. For the multi-class system, WAPITI's default configuration is applied to determine the label, i.e. label which has the highest score is assigned to word. In case of the binary system, the classification task is then conducted multiple times, corresponding to a threshold increase from 0.300 to 0.975 (step = 0.025). When threshold = α, all words in the test set which the probability of *"K"* class $\geqslant \alpha$ will be labelled as *"K"*, and otherwise, *"C"*. The values of Precision (Pr), Recall (Rc) and F-score (F) for *K* and *C* label are tracked along this threshold variation, allowing us to select the optimal threshold that yields the highest $F_{avg} = \frac{F(K)+F(C)}{2}$.

Results for the all-feature binary system (*ALL_BIN*) at the optimal threshold (0.500) and the multi-class one (*ALL_MULT*) at the default threshold, obtained on our dev set, are shown in Table 2. We can notice that with *ALL_BIN*, *"K"* label scores are very promising and *"C'* label reaches acceptable performance. In case of *ALL_MULT* we obtain the almost similar above performance for *"K"* and *"S"*, respectively, except the disappointing scores for *"D"* (which can be explained by the fact that very few instances of *"D"* words (4%) are observed in the training corpus).

Data set	**Train**	**Dev**	**Test**
#segments	753	50	284
#distinct segments	400	50	163
#words	18435	1306	7827
%K : %C	70: 30	77: 23	-
%K: %S: %D	70:26:4	77:19:4	-

Table 1: Statistics of training, dev and test sets

System	**Label**	**Pr(%)**	**Rc(%)**	**F(%)**
ALL_BIN	K	85.79	84.68	85.23
	C	50.96	53.16	52.04
ALL_MULT	K	85.30	84.00	84.65
	S	43.89	49.00	46.31
	D	7.90	6.30	7.01

Table 2: Average Pr, Rc and F for labels of all-feature binary and multi-class systems, obtained on dev set.

4 Feature Selection

In order to improve the preliminary scores of all-feature systems, we conduct a feature selection which is based on the hypothesis that some features may convey "noise" rather than "information" and might be the obstacles weakening the other ones. In order to prevent this drawback, we propose a method to filter the best features

[4]http://search.cpan.org/dist/Lingua-Wordnet/Wordnet.pm
[5]http://babelnet.org

based on the "Sequential Backward Selection" algorithm[6]. We start from the full set of N features, and in each step sequentially remove the most useless one. To do that, all subsets of (N-1) features are considered and the subset that leads to the best performance gives us the weakest feature (not involved in the considered set). This procedure is also called "leave one out" in the literature. Obviously, the discarded feature is not considered in the following steps. We iterate the process until there is only one remaining feature in the set, and use the following score for comparing systems: $F_{avg}(all) = \frac{F_{avg}(K)+F_{avg}(C)}{2}$, where $F_{avg}(K)$ and $F_{avg}(C)$ are the averaged F scores for K and C label, respectively, when threshold varies from 0.300 to 0.975. This strategy enables us to sort the features in descending order of importance, as displayed in Table 3. Figure 1 shows the evolution of the performance as more and more features are removed.

Rank	Feature name	Rank	Feature name
1	Source POS	14*	Distance to root
2*	Occur in Google Trans.	15	Backoff behaviour
3*	Nodes	16*	Constituent label
4	Target POS	17	Proper name
5	WPP	18	Number of occurrences
6	Left source context	19*	Min
7	Right target context	20*	Max
8	Numeric	21	Left target context
9*	Polysemy (target)	22*	Polysemy (source)
10	Punctuation	23*	Longest target gram length
11	Stop word	24*	Longest source gram length
12	Right source context	25	Source Word
13	Target Word		

Table 3: The rank of each feature (in term of usefulness) in the set. The symbol "*" indicates our proposed features.

Observations in 10-best and 10-worst performing features in Table 3 suggest that numerous features extracted directly from SMT system itself (source and target POS, alignment context information, WPP, lexical properties: numeric, punctuation) perform very well. Meanwhile, opposite from what we expected, those from word statistical knowledge sources (target and source language models) are likely to be much less beneficial. Besides, three of our proposed features appear in top 10-best. More noticeable, among them, the first-time-experimented feature "Occurrence in Google Translation hypothesis" is the most prominent (rank 2), implying that such an online MT system can be a reliable reference channel for predicting word quality.

[6]http://research.cs.tamu.edu/prism/lectures/pr/pr_l11.pdf

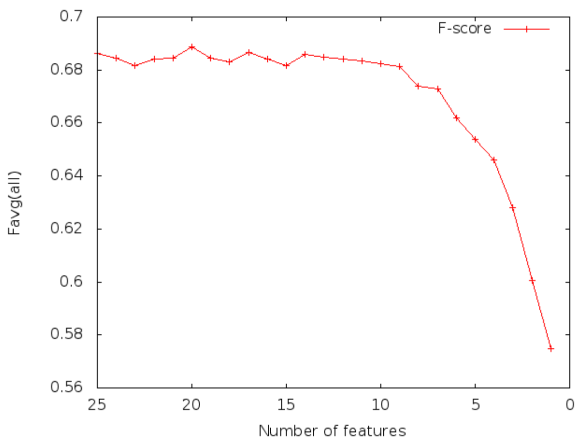

Figure 1: Evolution of system performance ($F_{avg}(all)$) during Feature Selection process, obtained on dev set

The above selection process also brings us the best-performing feature set (Top 20 in Table 3). The binary classifier built using this optimal subset of features (*FS_BIN*) reaches the optimal performance at the threshold value of 0.475, and slightly outperforms *ALL_BIN* in terms of F scores (0.46% better for *"K"* and 0.69% better for *"C"*). We then use this set to build the multi-class one (*FS_MULT*) and the results are shown to be a bit more effective compare to *ALL_MULT* (0.37% better for *"K"*, 0.80% better for *"S"* and 0.15% better for *"D"*). Detailed results of these two systems can be found in Table 4.

In addition, in Figure 1, when the size of feature set is small (from 1 to 7), we can observe sharply the growth of system scores for both labels. Nevertheless the scores seem to saturate as the feature set increases from the 8 up to 25. This phenomenon raises a hypothesis about the learning capability of our classifier when coping with a large number of features, hence drives us to an idea for improving the classification scores. This idea is detailed in the next section.

System	Label	Pr(%)	Rc(%)	F(%)
FS_BIN	K	85.90	85.48	85.69
	C	52.29	53.17	52.73
FS_MULT	K	85.05	85.00	85.02
	S	45.36	49.00	47.11
	D	9.1	5.9	7.16

Table 4: The Pr, Rc and F for labels of binary and multi-class system built from Top 20 features, at the optimal threshold value, obtained on dev set

5 Using Boosting technique to improve the system's performance

In this section, we try to answer to the following question: if we build a number of "weak" (or "basic") classifiers by using subsets of our features and a machine learning algorithm (such as *Boosting*), would we get a single "strong" classifier? When deploying this idea, our hope is that multiple models can complement each other as one feature set might be specialized in a part of the data where the others do not perform very well.

First, we prepare 23 feature subsets $(F_1, F_2, ..., F_{23})$ to train 23 basic classifiers, in which: F_1 contains all features, F_2 is the Top 20 in Table 3 and F_i $(i = \overline{3..23})$ contains 9 randomly chosen features. Next, a 7-fold cross validation is applied on our training set. We divide it into 7 subsets (S_1, S_2, $\ldots$, S_7). Each S_i $(i = \overline{1..6})$ contains 100 sentences, and the remaining 153 sentences constitute S_7. In the loop i $(i = \overline{1..7})$, S_i is used as the test set and the remaining data is trained with 23 feature subsets. After each loop, we obtain the results from 23 classifiers for each word in S_i. Finally, the concatenation of these results after 7 loops gives us the training data for Boosting. Therefore, the Boosting training file has 23 columns, each represents the output of one basic classifier for our training set. The detail of this algorithm is described below:

Algorithm to build Boosting training data

```
for i :=1 to 7 do
begin
    TrainSet(i) := $\cup S_k$ $(k = \overline{1..7}, k \neq i)$
    TestSet(i) := $S_i$
    for j := 1 to 23 do
    begin
        Classifier $C_j$ := Train TrainSet(i) with $F_j$
        Result $R_j$ := Use $C_j$ to test $S_i$
        Column $P_j$ := Extract the "probability of word
        to be G label" in $R_j$
    end
    Subset $D_i$ (23 columns) := $\{P_j\}$ $(j = \overline{1..23})$
end
Boosting training set $D$ := $\cup D_i$ $(i = \overline{1..7})$
```

Next, the Bonzaiboost toolkit[7] (which implements Boosting algorithm) is used for building Boosting model. In the training command, we invoked: algorithm = "AdaBoost", and number of iterations = 300. The Boosting test set is prepared as follows: we train 23 feature subsets with the **training set** to obtain 23 classifiers, then use them to test our **dev set**, finally extract the 23 probability columns (like in the above pseudo code). In the testing phase, similar to what we did in Section 4, the Pr, Rc and F scores against threshold variation for *"K"* and *"C"* labels are tracked, and those corresponding to the optimal threshold (0.575 in this case) are represented in Table 5.

System	Label	Pr(%)	Rc(%)	F(%)
BOOST_BIN	K	86.65	84.45	85.54
	C	51.99	56.48	54.15

Table 5: The Pr, Rc and F for labels of Boosting binary classifier (BOOST_BIN)

The scores suggest that using Boosting algorithm on our CRF classifiers' output accounts for an efficient way to make them predict better: on the one side, we maintain the already good achievement on *K* class (only 0.15% lost), on the other side we gain 1.42% the performance in *C* class. It is likely that Boosting enables different models to better complement each other, in terms of the later model becomes experts for instances handled wrongly by the previous ones. Another advantage is that Boosting algorithm weights each model by its performance (rather than treating them equally), so the strong models (come from all features, top 20, etc.) can make more dominant impacts than the rest.

6 Submissions and Official Results

After deploying several techniques to improve the system's prediction capability, we select two bests of each variant (binary and multi-class) to submit. For the binary task, the submissions include: the Boosting (BOOST_BIN) and the Top 20 (FS_BIN) system. For the multi-class task, we submit: the Top 20 (FS_MULT) and the all-feature (ALL_MULT) one. Before the submission, the training and dev sets were combined to re-train the prediction models for FS_BIN, FS_MULT and ALL_MULT. Table 6 reports the official results obtained by LIG at WMT 2013, task 2. We obtained the best performance among 3 participants. These results confirm that the feature selection strategy is efficient (FS_MULT slightly better than ALL_MULT) while the contribution of Boosting is unclear (BOOST_BIN better than FS_BIN if F-measure is considered but worse if Accuracy is considered - the difference is not significant).

[7]http://bonzaiboost.gforge.inria.fr/x1-20001

System	Pr	Rc	F	Acc
BOOST_BIN	0.777882	0.884325	0.827696	0.737702
FS_BIN	0.788483	0.864418	0.824706	0.738213
FS_MULT	-	-	-	0.720710
ALL_MULT	-	-	-	0.719177

Table 6: Official results of the submitted systems, obtained on test set

7 Discussion and Conclusion

In this paper, we describe the systems submitted for Task 2 of WMT13 Quality Estimation campaign. We cope with the prediction of quality at word level, determining whether each word is “good” or “bad” (in the binary variant), or is “good”, or should be “substitute” or “delete” (in the multi-class variant). Starting with the existing word features, we propose and add various of novel ones to build the binary and multi-class baseline classifier. The first experiment's results show that precision and recall obtained in *“K”* label (both in binary and multi-class systems) are very encouraging, and *“C”* (or *“S”*) label reaches acceptable performance. A feature selection strategy is then deployed to enlighten the valuable features, find out the best performing subset. One more contribution we made is the protocol of applying Boosting algorithm, training multiple “weak” classifiers, taking advantage of their complementarity to get a “stronger” one. These techniques improve gradually the system scores (measure with F score) and help us to choose the most effective systems to classify the test set.

In the future, this work can be extended in the following ways. Firstly, we take a deeper look into linguistic features of word, such as the grammar checker, dependency tree, semantic similarity, etc. Besides, we would like to reinforce the segment-level confidence assessment, which exploits the context relation between surrounding words to make the prediction more accurate. Moreover, a methodology to evaluate the sentence confidence relied on the word- and segment- level confidence will be also deeply considered.

References

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. Technical report, JHU/CLSP Summer Workshop, 2003.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. In *Proceedings of COLING 2004*, pages 315–321, Geneva, April 2004.

Mariano Felice and Lucia Specia. Linguistic features for quality estimation. In *Proceedings of the 7th Workshop on Statistical Machine Translation*, pages 96–103, Montreal, Canada, June 7-8 2012.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting et labeling sequence data. In *Proceedings of ICML-01*, pages 282–289, 2001.

Thomas Lavergne, Olivier Cappé, and François Yvon. Practical very large scale crfs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513, 2010.

Ngoc-Quang Luong. Integrating lexical, syntactic and system-based features to improve word confidence estimation in smt. In *Proceedings of JEP-TALN-RECITAL*, volume 3 (RECITAL), pages 43–56, Grenoble, France, June 4-8 2012.

Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411, Rochester, NY, April 2007.

S. Raybaud, D. Langlois, and K. Smaï li. ”this sentence is wrong.” detecting errors in machine - translated sentences. In *Machine Translation*, pages 1–34, 2011.

Radu Soricut and Abdessamad Echihabi. Trustrank: Inducing trust in automatic translations via ranking. In *Proceedings of the 48th ACL (Association for Computational Linguistics)*, pages 612–621, Uppsala, Sweden, July 2010.

Nicola Ueffing and Hermann Ney. Word-level confidence estimation for machine translation using phrased-based translation models. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 763–770, Vancouver, 2005.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. Confidence measures for statistical machine translation. In *Proceedings of the MT Summit IX*, pages 394–401, New Orleans, LA, September 2003.

Deyi Xiong, Min Zhang, and Haizhou Li. Error detection for statistical machine translation using linguistic features. In *Proceedings of the 48th Association for Computational Linguistics*, pages 604–611, Uppsala, Sweden, July 2010.