VNU Journal of Science: Comp. Science & Com. Eng. Vol. 30, No. 3 (2014) 36–49

Some Propositions to Improve the Prediction Capability of Word Confidence Estimation for Machine Translation

Ngoc Quang Luong, Laurent Besacier, Benjamin Lecouteux

Laboratoire d'Informatique de Grenoble,
41, Rue des Mathématiques, UJF - BP53, F-38041 Grenoble Cedex 9, France

Abstract

Word Confidence Estimation (WCE) is the task of predicting the correct and incorrect words in the MT output.test Dealing with this problem, this paper proposes some ideas to build a binary estimator and then enhance its prediction capability. We integrate a number of features of various types (system-based, lexical, syntactic and semantic) into the conventional feature set, to build our classifier. After the experiment with all features, we deploy a "Feature Selection" strategy to filter the best performing ones. Next, we propose a method that combines multiple "weak" classifiers to build a strong "composite" classifier by taking advantage of their complementarity. Experimental results show that our propositions helped to achieve a better performance in term of F-score. Finally, we test whether WCE output can play any role in improving the sentence level confidence estimation system.

Manuscript communication: received 15 December 2013, revised 04 April 2014, accepted 07 April 2014
Corresponding author: Luong Ngoc Quang, quangngocluong@gmail.com

Keywords: Machine Translation, Confidence Measure, Confidence Estimation, Conditional Random Fields, Boosting

1. Introduction

Statistical Machine Translation (SMT) systems in recent years have marked impressive breakthroughs with numerous commendable achievements, as they produced more and more user-acceptable outputs. Nevertheless the users still face with some open questions: are these translations ready to be published as they are? Are they worth to be corrected or do they require retranslation? It is undoubtedly that building a method which is capable of pointing out the correct parts as well as detecting the translation errors in each MT hypothesis is crucial to tackle these above issues. If we limit the concept "parts" to "words", the problem is called Word-level Confidence Estimation (WCE) [1].

The WCE's objective is to judge each word in the MT hypothesis as correct or incorrect by tagging it with an appropriate label. A classifier which has been trained beforehand calculates the confidence score for the MT output word, and then compares it with a pre-defined threshold. All words with scores that exceed this threshold are categorized in the *Good* label set; the rest belongs to the *Bad* label set.

The contributions of WCE for the other aspects of MT are incontestable. First, it assists the post-editors to quickly identify the translation errors [2], determine whether to correct the sentence or retranslate it from scratch, hence improve their productivity. Second, the confidence score of words is a potential clue to re-rank the SMT N-best lists [3, 2]. Last but not least, WCE can also be used by the translators in an interactive scenario [4].

This article integrates a number of our novel features into the conventional feature set and trains them by a *conditional random fields* (CRF)

model to build a classifier for WCE. We then set up a feature selection procedure, which identifies the most useful indicators for the prediction. Finally, we propose a method to improve the WCE performance by taking advantage of multiple sub-models' complementarity. In the next section, we review some previous researches about confidence estimation. Section 3 details the features used for the classifier construction. Section 4 lists our settings to prepare for the preliminary experiments and the baseline experimental results are reported in Section 5. Section 6 explains our feature selection procedure. Section 7 describes the Boosting method to improve the system performance. The integration of WCE into Sentence Confidence Estimation (SCE) system is presented in Section 8. The last section concludes the paper and points out some ongoing researches.

2. Related Work

To cope with WCE, various approaches have been proposed, aiming at two major issues: features and Machine Learning (ML) model to build the classifier. In this review, we refer mainly to two general types of features: internal and external features. "*Internal features*" (or "*system-based features*") are extracted from the components of MT system itself, generated before or during translation process (N-best lists, word graph, alignment table, language model, etc.). "*External features*" are constructed thanks to external linguistic knowledge sources and tools, such as Part-Of-Speech (POS) Tagger, syntactic parser, WordNet, stop word list, etc.

The authors in [5] combine a considerable number of features by applying neural network and naive Bayes learning algorithms. Among these features, Word Posterior Probability (henceforth WPP) proposed by [6] is shown to be the most effective system-based features. The combination of WPP (with 3 different variants) and IBM-Model 1 features is also shown to overwhelm all the other single ones, including heuristic and semantic features [7]. Using solely N-best list, the authors in [8] suggest 9 different features and then adopt a smoothed naive Bayes classification model to train the classifier.

Another study [1] introduces a novel approach that explicitly explores the phrase-based translation model for detecting word errors. A phrase is considered as a contiguous sequence of words and is extracted from the word-aligned bilingual training corpus. The confidence value of each target word is then computed by summing over all phrase pairs in which the target part contains this word. Experimental results indicate that the method yields an impressive reduction of the classification error rate compared to the state-of-the-art on the same language pairs.

In [9], the classifier is built by integrating the POS of the target word with another lexical feature named "Null Dependency Link" and training them by Maximum Entropy model. Interestingly, linguistic features sharply outperform WPP feature in terms of F-score and classification error rate.

Unlike most of previous work, the authors in [10] apply solely external features with the hope that their classifier can deal with various MT approaches, from statistical-based to rule-based. Given a MT output, the BLEU score is predicted by their regression model. Results show that their system maintains consistent performance across various language pairs.

A method to calculate the confidence score for both words and sentences relied on a feature-rich classifier is proposed by [2]. The novel features employed include source side information, alignment context, and dependency structure. Their integration helps to augment marginally in F-score as well as the Pearson correlation with human judgment. Moreover, their CE scores assist MT system to re-rank the N-best lists which improves considerably translation quality.

A recent study [11] applies 70 linguistic features guided by three main aspects of translation: accuracy, fluency and coherence to investigate their usefulness. Unfortunately these features were not yet able to beat shallower features based on statistics from the input text, its translation and additional corpora. Results

reveal that linguistic features are still helpful, but need to be carefully integrated to reach better performance.

In the submitted system to the WMT12 shared task on Quality Estimation, the authors in [12] add some new features to the baseline provided by the organizers, including averaged, intra-lingual, basic parser and out-of-vocabulary features. They are then trained by SVM model, then filtered by forward-backward feature selection algorithm. This algorithm waives features which linearly correlated with others while keeping those relevant for prediction. It increases slightly the performance of all-feature system in terms of Root Mean Square Error (RMSE).

Aiming at an MT system-independent quality assessment, "referential translation machines" (RTM) method proposed in [13] shows its prediction performance in WMT 2013, without accessing any SMT system specific resource and prior knowledge used to train data or model. RTM takes into account the acts of translation when translating between two data sets with respect to a reference corpus in the same domain.

Our work differs from previous researches at these main points: firstly, we integrate various types of prediction indicators: system-based features extracted from the MT system (N-best lists with the score of the log-linear model, source and target language model etc.), together with lexical, syntactic and semantic features to see if this combination improves the baselines performance [14]. Different from our previous work [14], this time we apply multiple ML models to train this feature set and then compare the performance to select the optimal one among them. Secondly, the usefulness of all features is deeper investigated in detail using a greedy feature selection algorithm. Thirdly, we propose a solution which exploits Boosting algorithm as a learning method in order to strengthen the contribution of dominant feature subsets to the system, thus improve of the system's prediction capability. Lastly, we explore the contribution of WCE in enhancing the quality estimation at sentence level. All these initiatives will be consequentially introduced, starting by the feature set building.

3. Features

This section depicts in details 25 features exploited to train our classifier. Among them, those marked with a Ⓟ symbol are proposed by us, and the remaining comes from the previous work. Interestingly, these features have been used in our English - Spanish WCE system which got the first rank in WMT 2013 Quality Estimation shared task (Task 2) [15].

3.1. System-based Features

They are the features extracted directly from our baseline SMT system, without the participation of any additional external component. Based on the resources where features are found, they can be sub-categorized as following:

3.1.1. Target Side Features

We take into account the information of every word (at position i in the MT output), including:

- The word itself
- The sequences formed between it and a word before ($i-1/i$) or after it ($i/i+1$)
- The trigram sequences formed by it and two previous and two following words (including: $i-2/i-1/i$; $i-1/i/i+1$; $i/i+1/i+2$)
- The number of occurrences in the sentence

3.1.2. Source Side Features

Using the alignment information, we can track the source words which the target word is aligned to. To facilitate the alignment representation, we apply the BIO[1] format: in case of multiple target words are aligned with one source word, the first word's alignment information will be prefixed with symbol "B-" (means "Begin");

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

Table 1. Example of using BIO format to represent the alignment information

Target words (MT output)	**Source aligned words**	**Target words (MT output)**	**Source aligned words**
The	B-le	to	B-de
public	B-public	look	B-tourner
will	B-aura	again	B-à|nouveau
soon	B-bientôt	at	B-son
have	I-aura	its	I-son
the	B-l'	attention	B-attention
opportunity	B-occasion	.	B-.

and "I-" (means "Inside") will be added at the beginning of alignment information for each of the remaining ones. The target words which are not aligned with any source word will be represented as "O" (means "Outside"). Table 1 shows an example for this representation, in case of the hypothesis is *"The public will soon have the opportunity to look again at its attention."*, given its source: *"Le public aura bientôt l'occasion de tourner à nouveau son attention."*. Since two target words *"will"* and *"have"* are aligned to *"aura"* in the source sentence, the alignment information for them will be "B-aura" and "I-aura" respectively. In case a target word has multiple aligned source words (such as *"again"*), we separate these words by the symbol "|" after putting the prefix "B-" at the beginning.

3.1.3. Alignment Context Features

These features are proposed by [2] in regard with the intuition that collocation is a believable indicator for judging if a target word is generated by a particular source word. We also apply them in our experiments, containing:

- 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.

For instance, in case of *"opportunity"* in Table 1, the source alignment context features are: *"opportunity/l"'* and *"opportunity/de"*; while the target alignment context features are: *"occasion/have"*, *"occasion/the"*, *"occasion/opportunity"*, *"occasion/to"*, *"occasion/look"*.

3.1.4. Word Posterior Probability

WPP [6] is the likelihood of the word occurring in the target sentence, given the source sentence. Numerous knowledge sources have been proposed to calculate it, such as word graphs, N-best lists, statistical word or phrase lexical. To calculate it, the key point is to determine sentences in N-best lists that contain the word e under consideration in a fixed position i . Let $p(f_1^J, e_1^I)$ be the joint probability of source sentence f_1^J and target sentence e_1^I. The WPP of e occurring in position i is computed by aggregating probabilities of all sentences containing e in this position:

$$p_i(e|f_1^J) = \frac{p_i(e, f_1^J)}{\sum_{e'} p_i(e', f_1^J)} \quad (1)$$

where

$$p_i(e, f_1^J) = \sum_{I, e_1^I} \Theta(e_i, e) \cdot p(f_1^J, e_1^I) \quad (2)$$

Here $\Theta(.,.)$ is the Kronecker function. The normalization in equation (1) is:

$$\sum_{e'} p_i(e', f_1^J) = \sum_{I, e_1^I} p(f_1^J, e_1^I) = p(f_1^J) \quad (3)$$

In this work, we exploit the graph that represents MT hypotheses [16]. From this, the WPP of word e in position i (denoted by WPP *exact*) can be calculated by summing up the probabilities of all paths containing an edge annotated with e in position i of the target sentence. Another form is "WPP *any*" in case we ignore the position i, or in other words, we sum up the probabilities of all paths containing an edge annotated with e in any position of the target sentence. Here, both forms are used and the above summation is performed by applying the forward-backward algorithm [17].

3.1.5. Graph topology features

They are based on the N-best list graph merged into a confusion network. On this network, each word in the hypothesis is labeled 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 (which the hypothesis word belongs to), we find some information that can be the 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*Ⓟ) . We assign three above numbers as features for the hypothesis word.

3.1.6. Language Model Based Features

Applying SRILM toolkit [18] with the bilingual corpus, we build 4-gram language models for both target and source side. These language models permit to compute the *"longest target n-gram length"*Ⓟ and *"longest source n-gram length"*Ⓟ (length of the longest sequence created by the current token and its previous ones in the target or source language model) of each word in MT output as well as in the source sentence. For example, with the target current token w_i: if the sequence $w_{i-2}w_{i-1}w_i$ appears in the target language model 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 value set for each word hence ranges from 0 to 4. Similarly, we compute the same value for the source word aligned to w_i in the source language model, and use both of them as features.

Additionally, we employ another feature named the *backoff behavior* [19] of the backward 3-gram target language model to investigate more deeply the role of two previous words by considering various cases of their occurrences, from which a score is given to each word w_i, as below:

$$B(w_i) = \begin{cases} 7 & \text{if } w_{i-2}, w_{i-1}, w_i \text{ exists} \\ 6 & \text{if } w_{i-2}, w_{i-1} \text{ and } w_{i-1}, w_i \text{ both exist} \\ 5 & \text{if only } w_{i-1}, w_i \text{ exists} \\ 4 & \text{if } w_{i-2}, w_{i-1} \text{ and } w_i \text{ exist separately} \\ 3 & \text{if } w_{i-1} \text{ and } w_i \text{ both exist} \\ 2 & \text{if only } w_i \text{ exists} \\ 1 & \text{if } w_i \text{ is out of vocabulary} \end{cases} \tag{4}$$

(The concept "exist" here means "appear in the language model").

3.2. Lexical Features

A prominent lexical feature that has been widely explored in WCE researches is word's Part-Of-Speech (POS). This tag is assigned to each word due to its syntactic and morphological behaviors to indicate its lexical category. We use TreeTagger[2] toolkit for POS annotation task and obtain the following features for each target word:

- Its POS
- Sequence of POS of all source words aligned to it (in BIO format)
- Bigram and trigram sequences between its POS and the POS of previous and following words. Bigram sequences are POS_{i-1}, POS_i and POS_i, POS_{i+1} and trigram sequences are: $POS_{i-2}, POS_{i-1}, POS_i$; $POS_{i-1}, POS_i, POS_{i+1}$ and $POS_i, POS_{i+1}, POS_{i+2}$

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

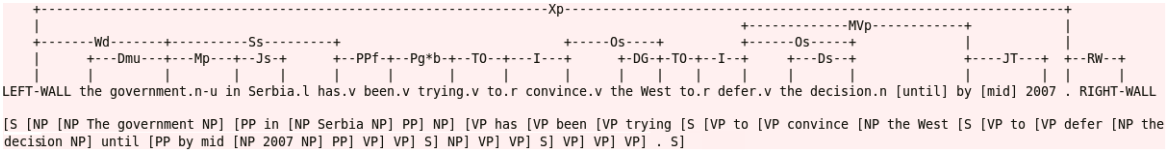


Fig. 1. Example of parsing result generated by Link Grammar

In addition, we also build four other binary 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*.

3.3. Syntactic Features

Besides lexical features, the syntactic information about a word is also a potential hint for predicting its correctness. If a word has grammatical relations with the others, it will be more likely to be correct than those which has no relation. In order to obtain the links between words, we select the Link Grammar Parser[3] as our syntactic parser, affording us to build a syntactic structure for each sentence in which each pair of grammar-related words is connected by a labeled link. In case of Link Grammar fails to find the full linkage for the whole sentence, it will skip each word one time until the sub-linkage for the remaining words has been successfully built. Based on this structure, we get the *"Null Link"* [9] characteristic of the word. This feature is binary: 0 in case of word has at least one link with the others, and 1 if otherwise. Another benefit yielded by this parser is the "constituent" tree (Penn tree-bank style phrase tree) representing the sentence's grammatical structure (showing noun phrases, verb phrases, etc.). This tree helps to produce more word syntactic features, including *its constituent label*Ⓟ and *its depth in the tree*Ⓟ (or the distance between it and the tree root).

It is intuitive to observe that the words in brackets (including "*until*" and "*mid*") have no link with the others, meanwhile the remaining ones have. For instance, the word "*trying*" is connected with "*to*" by the link "TO" and with "*been*" by the link "Pg*b". Hence, the value of "Null Link" feature for "*mid*" is 1 and for "*trying*" is 0. The figure also brings us the constituent label and the distance to the root of each word. In case of the word "*government*", these values are "NP" and "2", respectively.

3.4. Semantic Features

We study the semantic characteristic of word by taking into account its polysemy. We hope that the number of senses of each target word given its POS can be a reliable indicator for judging if it is the translation of a particular source word. The feature *"Polysemy count"*Ⓟ is built by applying a Perl extension named Lingua::WordNet[4], which provides functions for manipulating WordNet [5] database.

4. Experimental Settings

4.1. Our French - English SMT System

Our French - English SMT system is constructed using the Moses toolkit [20], which contains all of necessary components to train the translation model. We keep the Moses's default setting: log-linear model with 14 weighted feature functions. The translation model is trained on the Europarl and News parallel corpora used for WMT[6] evaluation campaign in 2010 (total 1,638,440 sentences). Our target language model is a standard n-gram language model trained using the SRI language modeling toolkit [18] on the news monolingual corpus (48,653,884 sentences). More details on this baseline system can be referred in [21].

[3]http://www.link.cs.cmu.edu/link/

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

[5]http://wordnet.princeton.edu/

[6]http://www.statmt.org/wmt10/

Table 2. Example of training label obtained using TERp-A.

Reference	The	consequence	of	the	fundamentalist	movement		also	has	its importance	.
		S			S	Y	I		D	P	
Hypothesis	The	result	of	the	hard-line	trend	is	also		important	.

4.2. Corpus Preparation

We use our SMT system to generate the translation hypothesis for 10,881 source sentences taken from news corpora of the WMT evaluation campaign (from 2006 to 2010). A post-edition task was implemented by using a crowdsourcing platform: Amazon Mechanical Turk (MTurk), which allows a *requester* to propose a paid or unpaid work and a *worker* to perform the proposed task. To avoid the gap between hypothesis and its post-edition since the correctors can paraphrase or reorder words to form the smoother translation, we highly recommend them to keep the number of edit operations as low as possible, but still ensure the accuracy of the translation. A sub-set (311 sentences) of these collected post-editions is then assessed by a professional translator. Testing result shows that 87.1% of post-editions improve the hypothesis. Detailed description for the corpus construction can be found in [22]. We extract 10,000 triples (source, hypothesis and post edition) to form the training set, and keep the remaining 881 triples for the test set.

4.3. Word Label Setting Using TERp-A

This task is performed by TERp-A toolkit [23]. As an extension of TER, TERp-A helps to eliminate its shortcomings by taking into account the linguistic edit operations, such as *Stem matches*, *Synonyms matches* and *Phrase Substitutions* besides the TER's conventional ones (*Exact match*, *Insertion*, *Deletion*, *Substitution* and *Shift*). These additions allow us to avoid categorizing the hypothesis word as Insertion or Substitution in case it shares same stem, or belongs to the same synonym set on WordNet, or is the phrasal substitution of word(s) in the reference. In TERp-A, each above-mentioned edit cost has been tuned to maximize the correlation with human judgment of Adequacy at the segment level. Table 2 illustrates the labels generated by TERp-A for one hypothesis and reference pair. Each word or phrase in the hypothesis is aligned to a word or phrase in the reference with discrepant types of edit: I (insertions), S (substitutions), T (stem matches), Y (synonym matches), and P (phrasal substitutions). The lack of a symbol indicates an exact match and will be replaced by E thereafter. We do not consider words marked with D (deletions) since they appear only in the reference. Then, to train a binary classifier, we re-categorize the obtained 6-label set into binary set: The E, T and Y are regrouped into the *Good* (G) category, whereas the S, P and I belong to the *Bad* (B) category. Finally, we observed that out of total words (train and test sets) are 85% labeled G, 15% labeled B.

4.4. Classifier Model Selection

In order to build the classifier, we train our features by several conventional models, such as: *Decision Tree* [24], *Logistic Regression* [25] and *Naive Bayes* [26] using KNIME platform[7]. However, since our intention is to treat WCE as a sequence labeling task, we employ also the CRF model [27]. Among CRF based toolkits, we selected WAPITI [28] to train our classifier. The training phase was conducted with Stochastic Gradient Descent (SGD) algorithm for L1-regularized model, which works by computing the gradient only on a single sequence at a time and making a small step in this direction, therefore it can quickly reach an acceptable solution for the model. In the training command, we set values for the maximum number of iterations (-maxiter), the stop window size (–stopwin) and the stop epsilon (–stopeps) to 200, 6 and 0.00005 respectively. We also

[7]http://www.knime.org/knime-desktop

compare our classifier with two naive baselines: in baseline 1, all words in each MT hypothesis are classified into *G* label. In baseline 2, we assigned them randomly into *G* or *B* with respect to the percentage between both labels in the corpus (85% *G*, 15% *B*).

5. Baseline WCE Experiments

We evaluate the performance of our classifiers by using common evaluation metrics: Precision (Pr), Recall (Rc) and F-score (F). Suppose that we would like to calculate these values for label B. Let X be the number of words whose true label is B and have been tagged with this label by the classifier, Y is the total number of words classified as B, and Z is the total number of words which true label is B. From these concepts, Pr, Rc and F can be defined as follows:

$$Pr = \frac{X}{Y}; Rc = \frac{X}{Z}; F = \frac{2 \times Pr \times Rc}{Pr + Rc} \quad (5)$$

These calculations can be applied in the same way for *G* label.

We perform our preliminary experiment by training a CRF classifier with the combination of all 25 features. The training algorithm and related parameters were discussed in Section 4.4. 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 for *G* class exceeds α will be labeled as "G", and the remaining will be labeled as "B". The values of Pr and Rc for *G* and *B* label are tracked along this threshold variation. The results observed show that in case of *B* label, Rc increases gradually from 0.285 to 0.492 whereas Pr falls from 0.438 to 0.353. With *G* label, the variation occurs in the opposite direction: Rc drops almost regularly from 0.919 to 0.799, meanwhile Pr augments slightly from 0.851 to 0.876.

Table 3 reports the *average* values of Precision, Recall and F-score of these labels in the all-feature system and the baseline systems (correspond to the above threshold variation).

Table 3. Average Precision, Recall and F-score for labels of all-feature system and two baselines.

System	Label	Pr(%)	Rc(%)	F(%)
All features	Good	85.99	88.18	87.07
	Bad	40.48	35.39	37.76
Baseline 1	Good	81.78	100.00	89.98
	Bad	-	0	-
Baseline 2	Good	81.77	85.20	83.45
	Bad	18.14	14.73	16.26

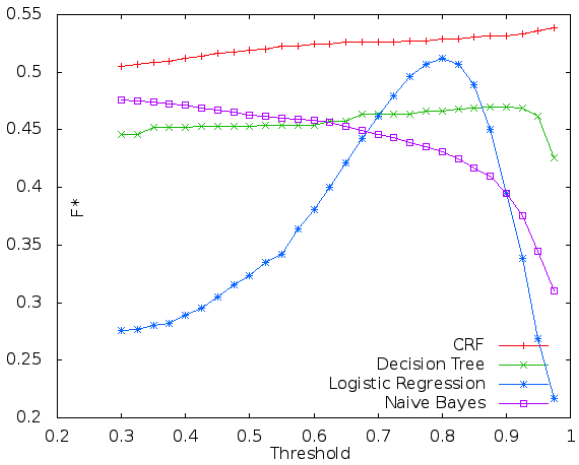


Fig. 2. Performance comparison (F^*) among different classifiers

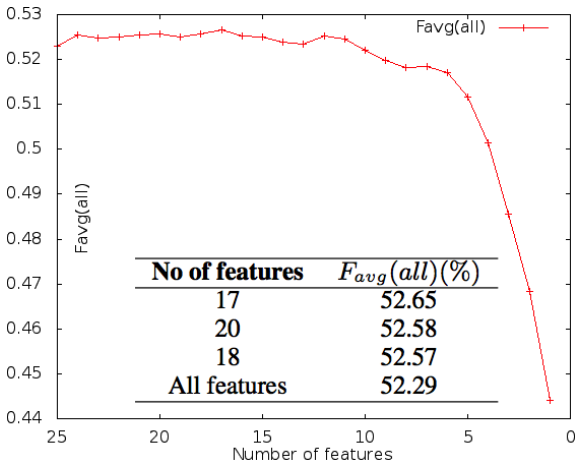
These values imply that in our system: (1) Good label is much better predicted than Bad label, (2) The combination of features helped to detect the translation errors significantly above the "naive" baselines.

In an attempt of investigating the performance of CRF model, we compare it with several other models, including: *Decision Tree*, *Logistic Regression* and *Naive Bayes*. These three classifiers are trained in the same condition (features, training set) of our CRF one, and then are used to test our usual test set. The pivotal problem is how to define an appropriate metric to compare them efficiently? Due to the fact that in our training corpus, the number of *G* words sharply beats the *B* ones, so it is fair to say that with our classifiers, detecting a translation error should be more appreciated than identifying a good translated word. Therefore, we propose a "composite" score called F^* putting more weight on the capability of each system in detecting translation error (represented by F-score for *B*

label). Specifically, this value can be written by: $F^* = 0.70 * Fscore(B) + 0.30 * Fscore(G)$. We track all scores along to the threshold variation and then plot them in Figure 2. The topmost position of CRF curve shown in the figure reveals that the CRF model performs better than all the remaining ones, and it is more suitable to deal with our features and corpus. Another notable observation is that the "optimal" threshold (which gives the best F^*) for each classifier is different from the others: 0.975 for *CRF*, 0.925 for *Decision Tree*, 0.800 for *Logistic Regression* and 0.300 for *Naive Bayes* classifier. In the next sections, which propose ideas to improve the prediction capability, we work only with the CRF classifier.

6. Feature Selection for WCE

In the previous section, the participation of all 25 features yielded promising F scores for *G* label, but not very convincing F scores for *B* label. That can be originated from the risk that not all of features are really useful, or in other words, some are poor predictors and might be the obstacles weakening the others combined with them. In order to prevent this drawback, we propose a method to filter the best features based on the "Sequential Backward Selection" algorithm[8]. 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 included 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: $F_{avg}(all) = 0.30 * F_{avg}(G) + 0.70 * F_{avg}(B)$, where $F_{avg}(G)$ and $F_{avg}(B)$ are the averaged F scores for *G* and *B* 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 4. In this table, the letter following each feature's ranking represents its category: "S" for system-based , "L" for lexical, "T" for syntactic, and "M" for semantic feature; and the symbol "*" (if possible) indicates that this is a our proposed feature. Figure 3 shows the evolution of the WCE performance as more and more features are removed; along with the details of 3 best-performing feature subsets yielding the highest F-scores.



No of features	$F_{avg}(all)(\%)$
17	52.65
20	52.58
18	52.57
All features	52.29

Fig. 3. Evolution of system performance ($F_{avg}(all)$) during Feature Selection process

Table 4 reveals that in our system, system-based and lexical features seemingly outperform the other types in terms of usefulness, since in top 10, they contribute 8 (5 system-based + 3 lexical). However, 2 out of 3 syntactic features appear in top 10, indicating that their role cannot be disdained. It is hard to conclude about the contribution of semantic feature because so far we have exploited only 1 representative and it ranks 15. Observation in 10-best and 10-worst performing features suggests that features belonging to word origin (word itself, POS) perform very well, meanwhile those from word statistical knowledge sources (target and source language models) are likely to be much less beneficial. More remarkable, we acknowledge the features which perform efficiently (appear in Top 10) both current system and in our English - Spanish one [15], including: *Source POS*, *Target Word*, *WPP (any)*, *Target*

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

Table 4. The rank of each feature (in term of usefulness) in the set

Rank	Feature name	Rank	Feature name
1L	Source POS	14L	Punctuation
2S	Source word	15M*	Polysemy count
3S	Target word	16S*	Longest source gram length
4S	Backoff behavior	17S	Number of occurrences
5S	WPP *any*	18L	Numeric
6L	Target POS	19L	Proper name
7T*	Constituent label	20S	Left target context
8S	Left source context	21S*	Min
9T	Null link	22S*	Longest target gram length
10L	Stop word	23S	Right source context
11S*	Max	24T*	Distance to root
12S	Right target context	25S	WPP *exact*
13S*	Nodes		

POS, and *Left source alignment context*. On the contrary, *"Left target alignment context"* and *"Longest target gram length"* perform poorly in both systems as they belong to top 5 at the bottom of the lists.

In addition, in Figure 3, 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.

7. Classifier Performance Improvement Using Boosting

As stated before, the best performance did not come from the "all-feature" system, but from the system trained by a subset of 17 features. Besides this, we could not find any considerable progression in F-score when the feature set is lengthened from 8 to 25. These observations lead to a question: If we build a number of "weak" (or "basic") classifiers by using subsets of our features, then train this classifier set by a machine learning algorithm (such as *Boosting* [29]), should we get a single "strong" classifier? If 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 sub feature sets $(F_1, F_2, ..., F_{23})$ to train 23 basic classifiers, in which:

- F_1 contains all features,
- F_2 contains 17 top-ranked in Table 4, and
- F_i $(i = \overline{3..23})$ contains 9 randomly chosen features.

Next, the 10-fold cross validation is applied on our usual 10K training set. We divide it into 10 equal subsets $(S_1, S_2, \ldots, S_{10})$. In the loop i $(i = \overline{1..10})$, S_i is used as the test set and the remaining data is trained with 23 sub feature sets. After each loop, we obtain the results from 23 classifiers for each word in S_i. Finally, the concatenation of these results after 10 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 CRF training set. The detail of this algorithm is described below:

Algorithm to build Boosting training data

for i :=1 **to** 10 **do**
begin

TrainSet(i) := $\cap S_j$ ($j = \overline{1..10}, j \neq i$)
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 $D := \cap D_i$ ($i = \overline{1..10}$)

Next, Bonzaiboost toolkit[9] (that is able to learn from decision trees and apply Boosting algorithm on them) is used for building Boosting model. In the training command, following parameters are invoked: algorithm = "AdaBoost", depth of the weak tree = 2, number of iterations = 300.

Meanwhile, the Boosting test set is prepared as follows: we train 23 feature sets with the usual 10K training set to obtain 23 classifiers, then use them to test the CRF test set, finally extract the 23 probability columns (like in the above pseudo code). In the testing phase, similar to what we did in Section 5, the *averaged* Pr, Rc and F scores against threshold variation for *G* and *B* labels are tracked as seen in Table 5. The distribution of F scores in both labels, compared to CRF system, is represented in Figure 4.

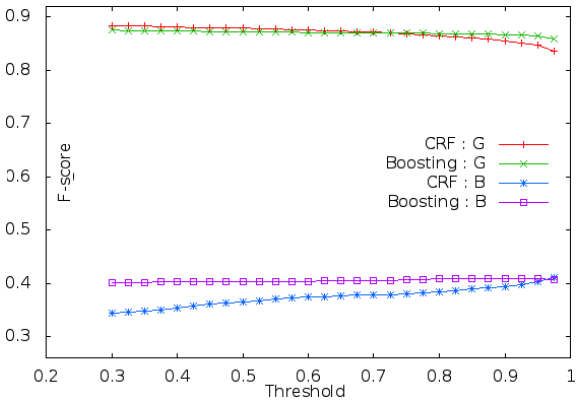


Fig. 4. Performance comparison between CRF and Boosting systems.

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

The scores suggest that using Boosting algorithm on our CRF classifiers' output is an efficient way to make them predict better: on the one side, we maintain the already good achievement on *G* class (only 0.05% lost), on the other side we augment 2.89% the performance in *B* class. It is likely that Boosting enables discrepant models to better complement one another, in terms of the later model becomes experts for instances handled wrongly by the previous ones. Another reason that yields the better score is that Boosting algorithm weights each model by its performance (rather than treating them equally), so the strong models (come from all features, top 17, etc.) can make more dominant impacts than the others. The results also show that all our features are helpful if they are carefully and skilfuly integrated.

8. Using WCE in Sentence Confidence Estimation (SCE)

WCE helps not only in detecting translation errors, but also in improving the sentence level prediction when combined with other sentence features. To verify this, firstly we build a SCE system (called **SYS1**) based on our WCE outputs (prediction labels). The seven features used to train **SYS1** are:

- The ratio of number of good words to total number of words. (1 feature)
- The ratio of number of good nouns to total number of nouns. The similar ones are also computed for other POS: verb, adjective and adverb. (4 features)
- The ratio of number of n consecutive good word sequences to total number of consecutive word sequences. Here, n=2 and n=3 are applied. (2 features)

Then, we inherit the script used in WMT12[10] for extracting 17 sentence features, to build an

[10]https://github.com/lspecia/QualityEstimation/blob/master/baseline_system

Table 5. Comparison of the average Pr, Rc and F between CRF and Boosting systems

System	Pr(G)	Rc(G)	F(G)	Pr(B)	Rc(B)	F(B)
Boosting	90.10	84.13	**87.02**	34.33	49.83	**40.65**
CRF (all)	85.99	88.18	**87.07**	40.48	35.39	**37.76**

another SCE system (**SYS2**). In both **SYS1** and **SYS2**, each sentence's training label is an integer score from 1 to 5, based on its TER score [23] (when matching against the post-edition), as following:

$$score(s) = \begin{cases} 5 & if\ TER(s) \leq 0.1 \\ 4 & if\ 0.1 \leq TER(s) \leq 0.3 \\ 3 & if\ 0.3 < TER(s) \leq 0.5 \\ 2 & if\ 0.5 < TER(s) \leq 0.7 \\ 1 & if\ TER(s) > 0.7 \end{cases} \quad (6)$$

Two conventional metrics are used to measure the SCE system's performance: Mean Absolute Error (MAE) and Root of Mean Square Error (RMSE)[11]. Given a test set $S = s_1, s_2, ..., s_{|S|}$, let $R(s_i)$ and $H(s_i)$ be the reference score (determined by TERPA) and hypothesis score (by our SCE system) for sentence s_i respectively. Then, MAE and RMSE can be formally defined by:

$$MAE = \frac{\sum_{i=1}^{|S|} |R(s_i) - H(s_i)|}{|S|} \quad (7)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{|S|} (|R(s_i) - H(s_i)|)^2}{|S|}} \quad (8)$$

To observe the impact of WCE on SCE, we design a third system (called **SYS1+SYS2**), which takes the results yielded by **SYS1** and **SYS2**, post-processes them and makes the final decision. For each sentence, **SYS1** and **SYS2** generate five probabilities for five integer labels it can be assigned, then select the label which highest probability as the official result. Meanwhile, **SYS1+SYS2** collects probabilities come from both systems, then updates the probability for each label by the sum of two appropriate values in **SYS1** and **SYS2**. Similarly, the label with highest likelihood is assigned to this sentence. The results obtained on the usual test set are shown in Table 6.

Table 6. Scores of 3 different SCE systems.

System	MAE	RMSE
SYS1	0.5584	0.9065
SYS2	0.5198	0.8707
SYS1+SYS2	*0.4835*	*0.8415*

Scores observed reveal that when WMT12 baseline features and those based on our WCE are separately exploited, they yield acceptable performance. More interesting, the contribution of WCE is definitively proven when it is combined with a SCE system: The combination system **SYS1+SYS2** sharply reduces MAE and RMSE of both single systems. It demonstrates that in order to judge effectively a sentence's overall quality, besides global and general indicators, the information synthesized from the quality of each word is also very useful.

9. Conclusions and Perspectives

We proposed some ideas to deal with WCE for MT, starting with the integration of our proposed features into the existing features to build the classifier. The first experiment's results show that precision and recall obtained in *G* label are very promising, and *B* label reaches acceptable performance. A feature selection strategy is then deployed to identify 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. All

[11] http://www.52nlp.com/mean-absolute-error-mae-and-mean-square-error-mse/

of these above propositions aim at enhancing the prediction capability for WCE. Finally, an investigation about the help of WCE in lifting up SCE system performance accentuates its increasing vital role in MT sectors.

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 investigate the segment-level confidence estimation, which exploits the context relation between surrounding words to make the prediction more accurate. Moreover, a methodology to conclude the sentence confidence relied on the word- and segment- level confidence will be deeply considered. We also plan to examine the WCE contribution for improving the MT quality, via various scenarios: N-best list re-ranking, Search Graph Re-decoding, etc.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to the earlier version of the paper.

References

[1] N. Ueffing, H. 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, Vancouver, 2005, pp. 763–770.

[2] B. Nguyen, F. Huang, Y. Al-Onaizan, Goodness: A method for measuring machine translation confidence, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Portland, Oregon, 2011, pp. 211–219.

[3] N. Q. Luong, L. Besacier, B. Lecouteux, Word confidence estimation for smt n-best list re-ranking, in: Proceedings of the Workshop on Humans and Computer-assisted Translation (HaCaT), Gothenburg, Sweden, 2014.

[4] S. Gandrabur, G. Foster, Confidence estimation for text prediction, in: Proceedings of the Conference on Natural Language Learning (CoNLL 2003), Edmonton, 2003, pp. 315–321.

[5] J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, N. Ueffing, Confidence estimation for machine translation, Tech. rep., JHU/CLSP Summer Workshop (2003).

[6] N. Ueffing, K. Macherey, H. Ney, Confidence measures for statistical machine translation, in: Proceedings of the MT Summit IX, New Orleans, LA, 2003, pp. 394–401.

[7] J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, N. Ueffing, Confidence estimation for machine translation, in: Proceedings of COLING 2004, Geneva, 2004, pp. 315–321.

[8] A. Sanchis, A. Juan, E. Vidal, Estimation of confidence measures for machine translation, in: Proceedings of the MT Summit XI, Copenhagen, Denmark, 2007, pp. 407–412.

[9] D. Xiong, M. Zhang, H. Li, Error detection for statistical machine translation using linguistic features, in: Proceedings of the 48th Association for Computational Linguistics, Uppsala, Sweden, 2010, pp. 604–611.

[10] R. Soricut, A. Echihabi, Trustrank: Inducing trust in automatic translations via ranking, in: Proceedings of the 48th ACL (Association for Computational Linguistics), Uppsala, Sweden, 2010, pp. 612–621.

[11] M. Felice, L. Specia, Linguistic features for quality estimation, in: Proceedings of the 7th Workshop on Statistical Machine Translation, Montreal, Canada, 2012, pp. 96–103.

[12] D. Langlois, S. Raybaud, K. S. li., Loria system for the wmt12 quality estimation shared task, in: Proceedings of the Seventh Workshop on Statistical Machine Translation, Association for Computational Linguistics, Montreal, Canada, 2012.

[13] E. Bicici, Referential translation machines for quality estimation, in: Proceedings of the Eighth Workshop on Statistical Machine Translation, Association for Computational Linguistics, Sofia, Bulgaria, 2013, pp. 343–351.

[14] N.-Q. Luong, Integrating lexical, syntactic and system-based features to improve word confidence estimation in smt, in: Proceedings of JEP-TALN-RECITAL, Vol. 3 (RECITAL), Grenoble, France, 2012, pp. 43–56.

[15] N. Q. Luong, B. Lecouteux, L. Besacier, LIG system for WMT13 QE task: Investigating the usefulness of features in word confidence estimation for MT, in: Proceedings of the Eighth Workshop on Statistical Machine Translation, Association for Computational Linguistics, Sofia, Bulgaria, 2013, pp. 396–391.

[16] N. Ueffing, F. J. Och, H. Ney, Generation of word graphs in statistical machine translation, in: Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP 02), Philadelphia, PA, 2002, pp. 156–163.

[17] N. Ueffing, H. Ney, Word-level confidence estimation for machine translation., in: Computational Linguistics, Vol. 33, 2007, pp. 9–40.

[18] A. Stolcke, Srilm - an extensible language modeling toolkit, in: Seventh International Conference on

Spoken Language Processing, Denver, USA, 2002, pp. 901–904.

[19] S. Raybaud, D. Langlois, K. Smaïli, "this sentence is wrong." detecting errors in machine-translated sentences., Machine Translation 25 (1) (2011) 1–34.

[20] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst, Moses: Open source toolkit for statistical machine translation, in: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 177–180.

[21] M. Potet, L. Besacier, H. Blanchon, The lig machine translation system for wmt 2010, in: A. Workshop (Ed.), Proceedings of the joint fifth Workshop on Statistical Machine Translation and Metrics MATR (WMT2010), Uppsala, Sweden, 2010.

[22] M. Potet, R. Emmanuelle E, L. Besacier, H. Blanchon, Collection of a large database of french-english smt output corrections, in: Proceedings of the eighth international conference on Language Resources and Evaluation (LREC), Istanbul, Turkey, 2012.

[23] M. Snover, N. Madnani, B. Dorr, R. Schwartz, Terp system description, in: MetricsMATR workshop at AMTA, 2008.

[24] J. R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106. doi:10.1023/A:1022643204877.

[25] J. Friedman, T. Hastie, R. Tibshirani, Additive Logistic Regression: a Statistical View of Boosting, The Annals of Statistics 38 (2).

[26] D. Lowd, Naive bayes models for probability estimation, in: Proceedings of the Twentysecond International Conference on Machine Learning, ACM Press, 2005, pp. 529–536.

[27] J. Lafferty, A. McCallum, F. Pereira, Conditional random fields: Probabilistic models for segmenting et labeling sequence data, in: Proceedings of ICML-01, 2001, pp. 282–289.

[28] T. Lavergne, O. Cappé, F. Yvon, Practical very large scale crfs, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 2010, pp. 504–513.

[29] R. E. Schapire, The boosting approach to machine learning: An overview (2002).