Word Confidence Estimation and its Integration in Sentence Quality Estimation for Machine Translation

Ngoc-Quang Luong, Laurent Besacier, Benjamin Lecouteux

Word Confidence Estimation and its Integration in Sentence Quality Estimation for Machine Translation

Ngoc-Quang Luong, Laurent Besacier, and Benjamin Lecouteux

Laboratoire d'Informatique de Grenoble, Campus de Grenoble
41, Rue des Mathématiques, BP53, F-38041 Grenoble Cedex 9, France
{Ngoc-Quang.Luong,Laurent.Besacier,Benjamin.Lecouteux}@imag.fr
http://www.liglab.fr

Abstract. This paper proposes some ideas to build an effective estimator, which predicts the quality of words in a Machine Translation (MT) output. 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. Once having experiments with all features, we deploy a "Feature Selection" strategy to filter the best performing ones. Then, a method that combines multiple "weak" classifiers to build a strong "composite" classifier by taking advantage of their complementarity allows us achieve a better performance in term of F score. Finally, we exploit word confidence scores for improving the estimation system at sentence level.

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 fruitful 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). 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, 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. Last but not least, WCE can also be used by the translators in an interactive scenario [2].

This article conveys ideas towards a better word quality prediction, including: novel features integration, feature selection and Boosting technique. It also investigates the usefulness of using WCE in a sentence-level confidence estimation (SCE) system. After reviewing some related researches in Section 2, we depict all the features used for the classifier construction in Section 3. The settings and results of our preliminary experiments are reported in Section 4. Section 5 explains our feature selection procedure. Section 6 describes the Boosting method to improve the system performance. The role of WCE in SCE is discussed in Section 7. The last section concludes the paper and points out some perspectives.

2 Previous Work Review

To cope with WCE, various approaches have been proposed, aiming at two major issues: features and model to build the classifier. In [1], the authors combine a considerable number of features, then train them by the Neural Network and naive Bayes learning algorithms. Among these features, Word Posterior Probability (henceforth WPP) proposed by [3] is shown to be the most effective system-based features. Moreover, its combination with IBM-Model 1 features is also shown to overwhelm all the other ones, including heuristic and semantic features [4].

A novel approach introduced in [5] 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.

Xiong et al. [6] integrate the POS of the target word with another lexical feature named "Null Dependency Link" and train them by Maximum Entropy model. In their results, linguistic features sharply outperform WPP feature in terms of F-score and classification error rate. Similarly, 70 linguistic features guided by three main aspects of translation: accuracy, fluency and coherence are applied in [9]. Results reveal that these features are helpful, but need to be carefully integrated to reach better performance.

Unlike most of previous work, the authors in [7] 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.

Nguyen et al. [8] study a method to calculate the confidence score for both words and sentences relied on a feature-rich classifier. 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.

3 Features

This section depicts in detail 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.

3.1 System-based Features (directly extracted from SMT system)

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: $i-2/i-1/i$; $i-1/i/i+1$; and $i/i+1/i+2$).
- The number of occurrences in the sentence.

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: if multiple target words are aligned with one source word, the first word's alignment information will be prefixed with symbol "B-" (means "Begin"); meanwhile "I-" (means "Inside") will be added at the beginning of the 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

Target words	Source aligned words	Target words	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-.

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

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.

Alignment Context Features These features are proposed by [8] 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.

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

– 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"* and *"occasion/look"*.

Word Posterior Probability WPP [3] is the likelihood of the word occurring in the target sentence, given the source sentence. 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. In this work, we exploit the graph that represents MT hypotheses [10]. 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 sum up the probabilities of all paths containing an edge annotated with e in any position of the target sentence. In this paper, both forms are employed.

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 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 (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.

Language Model Based Features Applying SRILM toolkit [11] on the bilingual corpus, we build 4-gram language models for both target and source side, which permit to compute two features: the *"longest target n-gram length"*Ⓟ and *"longest source n-gram length"*Ⓟ (length of the longest sequence created by the current word and its previous ones in the language model). For example, with the target word 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 word aligned to w_i in the source language model. Additionally, we consider also the *backoff behaviour* [18] of the target language model to the word w_i, according to how many time it has to back-off in order to assign a probability to the word sequence.

3.2 Lexical Features

A prominent lexical feature that has been widely explored in WCE researches is word's Part-Of-Speech (POS). We use TreeTagger[2] toolkit for POS annotation task and obtain the following features for each target word:

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

- Its POS
- Sequence of POS of all source words aligned to it
- 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 $POS_i, POS_{i+1}, POS_{i+2}$

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

The syntactic information about a word is 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, allowing us to build a syntactic structure for each sentence in which each pair of grammar-related words is connected by a labeled link. Based on this structure, we get a binary feature called *"Null Link"*: 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, 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).

Figure 1 represents the syntactic structure as well as the constituent tree for a MT output: *"The government in Serbia has been trying to convince the West to defer the decision until by mid 2007."*. It is intuitive to observe that the words in

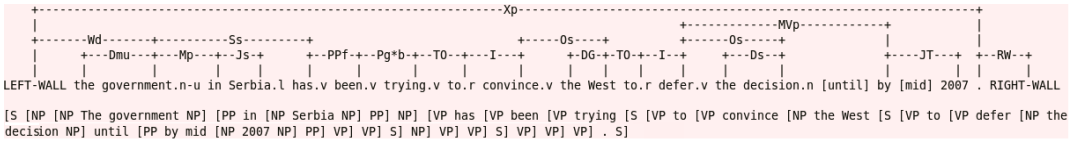


Fig. 1. Example of parsing result generated by Link Grammar

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

The word semantic characteristic that we study is 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*

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

count"Ⓟ is built by applying a Perl extension named Lingua::WordNet[4], which provides functions for manipulating WordNet database.

4 Baseline WCE Experiments

4.1 Experimental Settings

SMT System Our French - English SMT system is constructed using the Moses toolkit [12]. 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 WMT10[5] evaluation campaign (1,638,440 sentences). Our target language model is a standard n-gram language model trained using the SRI language modeling toolkit [11] on the news monolingual corpus (48,653,884 sentences). More details on this system can be found in [13].

Corpus Preparation We used our SMT system to obtain the translation hypothesis for 10,881 source sentences taken from news corpora of the WMT evaluation campaign (from 2006 to 2010). Our post-editions were generated by using a crowdsourcing platform: Amazon Mechanical Turk [14]. 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.

Word Label Setting This task is performed by TERp-A toolkit [15]. 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 different 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 the 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 belong to the *Good* (G), 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.

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

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

Classifier Selection We apply several conventional models, such as: *Decision Tree*, *Logistic Regression* and *Naive Bayes* using KNIME platform[6]. However, since our intention is to treat WCE as a sequence labeling task, we employ also the *Conditional Random Fields* (CRF) model [16]. Among CRF based toolkits, we selected WAPITI [17] to train our classifier. We also 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: 85% into *G* and 15% into *B* label (similar to the percentage of these labels in the corpus).

[4] http://search.cpan.org/dist/Lingua-Wordnet/Wordnet.pm
[5] http://www.statmt.org/wmt10/
[6] http://www.knime.org/knime-desktop

4.2 Preliminary Results and Analysis

We evaluate the performance of our classifier by using common evaluation metrics: Precision (Pr), Recall (Rc) and F-score (F). We perform our preliminary experiment by training a CRF classifier with the combination of all 25 features. 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 labelled as "G", and otherwise, "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

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

Table 3. Average Pr, Rc and F for labels of all-feature system and two baselines.

these labels in the all-feature system and the baseline systems (correspond to the above threshold variation). 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

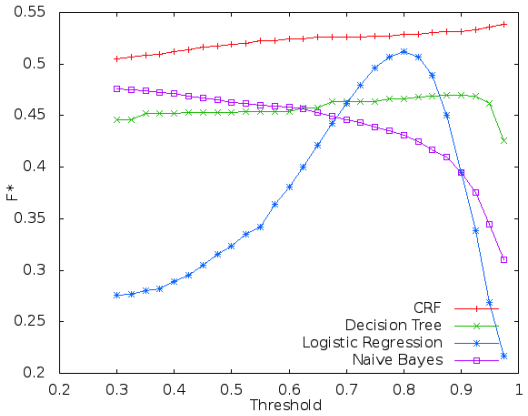


Fig. 2. Performance comparison (F^*) among different classifiers

with several other models, as stated in Section 4.1. 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 outperforms 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 system capability of detecting B words: $F^* = 0.70 * Fscore(B) + 0.30 * Fscore(G)$. 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.

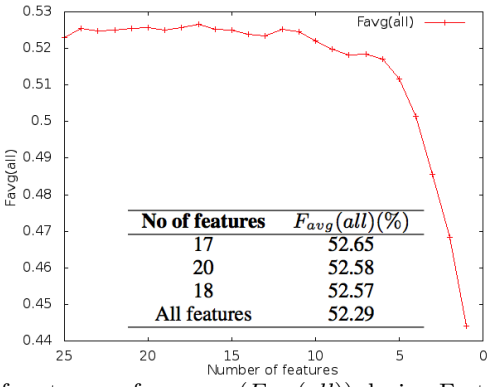
5 Feature Selection for WCE

In Section 4, the all-feature system 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 other ones. In order to prevent this drawback, we propose a method to filter the best features based on the "Sequential Backward Selection" algorithm[7]. 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). 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 by descending order of importance, as displayed in Table 4. Figure 3 shows the evolution of the WCE performance as more and more features are removed, and the details of 3 best feature subsets yielding the highest F-scores.

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 behaviour	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		

Table 4. The rank of each feature (in term of usefulness) in the set. The letter represents category: "S" for system-based , "L" for lexical, "T" for syntactic, and "M" for semantic feature; and the symbol "*" indicates our proposed features.

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



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. 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 much less beneficial. In addition, in Figure 3, when the size of feature set is small (from 1 to 7), we can observe sharply the growth of the system performance ($F_{avg}(all)$). Nevertheless the scores seem to saturate as the feature set increases from 8 up to 25. This phenomenon raises a hypothesis about our classifier's learning capability 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.

6 Classifier Performance Improvement Using Boosting

The results and observations in Section 5 lead to a 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*), 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 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 17 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, ..., 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 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 10 loops gives us the training data for Boosting. 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[8] (based on decision trees and 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 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

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**

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

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 different models to better complement one another, 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 17, etc.) can make more dominant impacts than the others.

7 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[9] for extracting 17 sentence features, to build an another SCE system (SYS2). In both SYS1 and SYS2, each sentence

[8] http://bonzaiboost.gforge.inria.fr/x1-20001
[9] https://github.com/lspecia/QualityEstimation/blob/master/baseline_system

training label is an integer score from 1 to 5, based on its TER score, as following:

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

Two conventional metrics are employed to measure the SCE system's performance: Mean Absolute Error (MAE) and Root of Mean Square Error (RMSE)[10]. 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 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 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 are shown in Table 6.

Scores observed reveal that when WMT12 baseline features and those based

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

Table 6. Scores of 3 different SCE systems.

on our WCE are separately exploited, they yield acceptable performance. And, their contributions are definitively proven when they are combined with WMT12 features: The combination system SYS1+SYS2 sharply reduces MAE and RMSE of both single systems. It demonstrates that in order to judge effectively a sentence, besides global and general indicators, information synthesized from the quality of each word are very useful.

8 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. Especially, the integration with SCE enlightens the WCE contribution in judging the sentence quality.

In the future, we will 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

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

accurate. Moreover, a methodology to conclude the sentence confidence relied on the word- and segment- level confidence will be also deeply considered.

References

1. Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis A., Ueffing, N.: Confidence Estimation for Machine Translation. Technical report, JHU/CLSP Summer Workshop (2003)
2. Gandrabur, S., Foster, G.: Confidence Estimation for Text Prediction. In: Conference on Natural Language Learning (CoNLL), pp. 315-321, Edmonton, May (2003)
3. Ueffing, N., Macherey, K., Ney, H.: Confidence Measures for Statistical Machine Translation. In: MT Summit IX, pp. 394-401, New Orleans, LA, September (2003)
4. Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., Ueffing, N.: Confidence Estimation for Machine Translation. In Proceedings of COLING 2004, pp. 315321, Geneva, April (2004)
5. Ueffing, N., Ney, H.: Word-level Confidence Estimation for Machine Translation Using Phrased-based Translation Models. In: Human Language Technology Conference and Conference on Empirical Methods in NLP, pp. 763-770, Vancouver (2005)
6. Xiong, D., Zhang, M., Li, H.: Error Detection for Statistical Machine Translation Using Linguistic Features. In: 48th Association for Computational Linguistics, pp. 604-611, Uppsala, Sweden, July (2010)
7. Soricut, R., Echihabi, A.: Trustrank: Inducing Trust in Automatic Translations via Ranking. In: 48th ACL (Association for Computational Linguistics), pp. 612-621, Uppsala, Sweden, July (2010)
8. Nguyen, B., Huang, F., Al-Onaizan, Y.: Goodness: A Method for Measuring Machine Translation Confidence. In: 49th Annual Meeting of the Association for Computational Linguistics, pp. 211-219, Portland, Oregon, June (2011)
9. Felice, M., Specia, L.: Linguistic Features for Quality Estimation. In: 7th Workshop on Statistical Machine Translation, pp. 96-103, Montreal, Canada, June 7-8 (2012)
10. Ueffing, N., Och, F.J., Ney, H.: Generation of Word Graphs in Statistical Machine Translation. In: Conference on Empirical Methods for Natural Language Processing (EMNLP 02), pp. 156-163, Philadelphia, PA (2002)
11. Stolcke, A.: Srilm - an Extensible Language Modeling Toolkit. In: 7th International Conference on Spoken Language Processing, pp. 901-904, Denver, USA (2002)
12. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open source toolkit for statistical machine translation. In: 45th Annual Meeting of the Association for Computational Linguistics, pp. 177-180, Prague, Czech Republic, June (2007)
13. Potet, M., Besacier, L., Blanchon, H.: The LIG Machine Translation System for WMT 2010. In: WMT2010, ACL Workshop. Uppsala, Sweden. 11-17 July (2010)
14. Potet, M., Rodier, E.E., Besacier, L., Blanchon, H.: Collection of a Large Database of French-English SMT Output Corrections. In: 8th International Conference on Language Resources and Evaluation, Istanbul (Turkey), 23-25 May (2012)
15. Snover, M., Madnani, N., Dorr, B., Schwartz, R.: Terp System Description. In: MetricsMATR workshop at AMTA (2008)
16. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: CML-01, pp. 282-289 (2001)
17. Lavergne, T., Cappé, O., Yvon, F.: Practical Very Large Scale CRFs. In: 48th Annual Meeting of the Association for Computational Linguistics, pp. 504-513 (2010)
18. Raybaud, S., Langlois, D., Smaili, K.: This sentence is wrong. Detecting errors in machine- translated sentences. Machine Translation, 25(1):1 34 (2011).