Multi-Classifier Combination for Translation Error Detection

Jinhua Du[1], Junbo Guo[2], Sha Wang[1], and Xiyuan Zhang[1]

[1] Faculty of Automation and Information Engineering,
jhdu@xaut.edu.cn
[2] Faculty of Advanced Technology,
Xi'an University of Technology, Xi'an, China

Abstract. This paper proposes a multi-classifier combination strategy to improve translation error detection performance for statistical machine translation (SMT). Specifically, two different classifiers – Maximum Entropy (MaxEnt) and Support Vector Machine (SVM) – over different features perform a binary classification and export classification probabilities for either class. Then a probability product rule based multi-classifier combination strategy is employed to fuse these two classifiers to decrease the classification error rate (CER). Three typical word posterior probabilities (WPP) and three linguistic features as well as their combinations are used in the experiments conducted on Chinese-to-English NIST data sets. Experimental results show that the combination of multiple classifiers reduce the CER by relative 0.15%, 0.94%, and 1.52% compared to the SVM classifier, and relative 1.73%, 1.72%, 2.02% compared to the MaxEnt classifier over three different feature combinations.

Keywords: translation error detection, binary classification, multi-classifier combination

1 Introduction

In recent years, a number of different types of SMT methods have been proposed, such as the phrase-based, hierarchical phrased-based, and syntax-based models etc., which significantly improve the translation quality. Meanwhile, a lot of effort has been put to apply SMT systems to practical use, e.g. the software localization industry [1–3]. However, the translation quality cannot fully satisfy the actual demand of industry yet. For example, the ungrammatical errors and disordered words in the translation often increase human cost. Therefore, high-quality automatic translation error detection or word-level confidence estimation is necessary to further improve the working efficiency of the post-editors or translators in the localization industry.

Typically, most translation error detection methods utilize system-based features (e.g. WPP) combining with extra knowledge such as linguistic features to decrease the classification error rate [4–9]. As to the system-based features, a number of different algorithms to calculate the WPP were proposed based on the N-best list or word lattice, and had been applied to SMT translation quality estimation. Afterwards, some researchers try to introduce more useful knowledge sources such as syntactic and semantic features to further improve the error detection capability [8, 10, 11]. However, these features are not that easy to extract due to their complexity, low generalization capability, and dependency on specific languages etc. Hence, currently the system-based features such as WPP and lexicalized features (e.g. word and part-of-speech (POS)) still play the main role in the error detection task or the confidence estimation task.

Generally, translation error detection can be regarded as a binary classification task. Thus, the accuracy of the classifier also plays an important role in terms of improving the prediction capability besides adding new features and extra knowledge. This paper mainly focuses on the investigation of different classifiers, and presents an effective and straightforward strategy of combining two different classifiers to improve the classification performance. Firstly, we introduce the features used in our task, which are three typical WPP system-based features and three linguistic features, then employ two different classifiers, namely the MaxEnt classifier and SVM classifier to perform the classification task respectively. Finally, we carry out a combination operation – multiplication of the classification probabilities – to obtain the final result. Experiments are conducted on NIST Chinese-to-English translation task, and the results show that the combined method outperforms either individual classifier used in our task in terms of the CER.

The rest of the paper is organized as follows: Section 2 briefs the related work as to the error detection task. In Section 3, three typical WPP and three linguistic features are described. Section 4 firstly describes the MaxEnt and SVM classifiers used in our task, and then the multi-classifier strategy and feature representation are detailed. Experimental settings, implementation and analysis are reported in Section 5. The final section concludes and gives avenues for future work.

2 Related Work

The question of translation confidence estimation has attracted a number of researcher due to its importance in promoting SMT application. In 2004, Blatz et al. improved the basic confidence estimation method by combining the neural network and a naive Bayes classifier to predict the word-level and the sentence-level translation errors [6]. The features they used include WPP calculated from the N-best list, translation model-based features, semantic feature extracted from the WordNet, as well as simple syntactic features. Experimental results show that all among these features, WPP is more effective with strong generalization capability than linguistic features.

Ueffing and Ney exhaustively explore various kinds of WPP features to perform confidence measures, and proposed different WPP algorithms to verify the effectiveness in confidence estimation task [5, 7]. In their task, the words in the generated target sentence can be tagged as *correct* or $false$ to facilitate post-editing or work in an interactive translation environment. Their experiments conducted on different data sets show that different WPP algorithms perform differently, but basically each can reduce the CER. Furthermore, the combination of different features can perform better than any individual features.

Specia et al. have done a lot of work with regard to the confidence estimation in the computer-aided translation field [10, 11]. They categorize translations into "bad" or "good" classes based on sentence-level binary scores of the post-edition MT fragments. The features used are called "black-box" features, which can be extracted from any MT systems only if the information from the input (source) and translation (target) sentences are given, such as source and target sentence lengths and their ratios, the edit distance between the source sentence and sentences in the corpus used to train the SMT system. Their work contributed significantly to SMT translation confidence estimation research and application.

Xiong et al. proposed an MaxEnt classifier based error detection method to predict translation errors (each word is tagged as *correct* or *incorrect*) by integrating a WPP feature, a syntactic feature extracted from LG parser and some lexical features [8]. The experimental results show that linguistic features can reduce CER when used alone, and it outperforms WPP. Moreover, linguistic features can further provide complementary information when combined with WPP, which collectively reduce the classification error rate.

In 2011, Bach et al. classified translation errors into four categories by extracting more richer set of source-side information features, and combined sentence -level and word-level features to estimate translation quality. They predict error types of each word in the MT output with a confidence score, then extend it to the sentence level, and finally apply it to N-best list re-ranking task to improve MT quality [9].

On the basis of previous work, this paper mainly focuses on how to significantly improve the classification performance by using different kinds of classifiers and combining multiple classifiers over a set of effective features. Specifically, this paper 1) verifies the performance of various classifiers, namely the MaxEnt classifier and the SVM classifier on the translation error detection task; 2) presents a probability product combination strategy to fuse two classifier to obtain better results.

3 Features

3.1 WPP Feature

WPP is served as a major and effective confidence estimation feature both in speech recognition and in SMT post-processing. In terms of SMT, WPP refers to the probability of a word occurring in the hypothesis given a source input. Generally speaking, the underlying idea is that if the posterior probability of a word occurring in a hypothesis is high, then the chance that it is believed to be correct is big correspondingly. Based on this consideration, it is reasonable that the more useful information considered in the WPP algorithm, the better the performance would achieve.

The general mathematical description of WPP is as:

For an SMT system S, given the input sentence f_1^J, and the exported N-best list $e_{n,1}^{n,I_n}$, where $n = 1, \ldots, N$, e_n refers to the n^{th} hypothesis with the probability $p(f_1^J, e_{n,1}^{n,I_n})$, then the WPP in the error detection task can be represented as calculating the probability $p_i(e|f_1^J, e_1^I)$ of the word e at position i in the 1-best hypothesis of the N-best list as in (1),

$$p_i(e|f_1^J, e_1^I) = \frac{\sum_{n=1}^{N} f(a, e_{n,i}, e) \cdot p(f_1^J, e_{n,1}^{n,I_n})}{\sum_{n=1}^{N} p(f_1^J, e_{n,1}^{n,I_n})} \tag{1}$$

where a is a hidden variable which indicates an alignment measure; $f(a, e_{n,i}, e)$ is a binary sign function as in (2),

$$f(a, e_{n,i}, e) = \begin{cases} 1 & e_{n,i} = e \\ 0 & otherwise \end{cases} \tag{2}$$

It can be seen from the description of N-best based WPP algorithm that the posterior probability of a word in a hypothesis can be worked out according to the sentence-level posterior probabilities of hypotheses in the N-best list. The vital information to be considered is the position of the word e which is determined by the alignment measure between the 1-best hypothesis and the rest of the N-best list.

Here we introduces three typical WPP methods to illustrate their different influence on the error detection performance over different kinds of classifiers.

3.1.1 Fixed Position based WPP

The basic idea of fixed position-based WPP is that given an input f_1^J, the posterior probability of a word e at position i in the hypothesis e_1^I can be calculated by summing the posterior probabilities of all sentences in the N-best list containing target word e at target position i, which is as in (3),

$$p_i(e|f_1^J, e_1^I) = \frac{\sum_{n=1}^{N} \delta(e_{n,i}, e) \cdot p(f_1^J, e_{n,1}^{n,I_n})}{\sum_{e'} \sum_{n=1}^{N} \delta(e_{n,i}, e') \cdot p(f_1^J, e_{n,1}^{n,I_n})} \tag{3}$$

where $\delta(x, y)$ is the Kronecker function as in (4),

$$\delta(x, y) = \begin{cases} 1 & x = y \\ 0 & otherwise \end{cases} \tag{4}$$

This method only uses the original position information of each word without any extra alignment measure between the 1-best and any other hypotheses.

3.1.2 Flexible Position based WPP

The potential problem of fixed position based WPP is that generally the hypotheses in the N-best list have different length that will make the same word occur at different positions so that the WPP would have a large error compared to the real probability distribution. Naturally the intuition to improve this method is to make the position flexible, e.g. using a sliding window.

The basic idea of sliding window is to consider the words around the position i, i.e., the context. Let the window size be t, then the sliding window at position i can be denoted as $i \pm t$. If the target word e appears inside the window, then we regard it occurring at position i and sum up the probability of the current hypothesis, which is formulated as in (5),

$$p_{i,t}(e|f_1^J, e_1^I) = \sum_{k=i-t}^{i+t} p_k(e|f_1^J, e_1^I) \tag{5}$$

where $p_k(e|f_1^J, e_1^I)$ is as illustrated in Eq. (3).

3.1.3 Word Alignment based WPP

The sliding window based method needs to choose a proper window size which can only be determined by experiments. Thus, another straightforward way to improve the fixed position method is to perform the word alignment between the 1-best hypothesis and the rest of hypotheses in the N-best list, i.e., align the rest of hypotheses against the 1-best hypothesis.

Specifically, let $L(e_1^I, e_{n,1}^{n,I_n})$ be the Levenshtein alignment between e_1^n and other hypotheses, then the WPP of the word e at position i is as in (6):

$$p_{lev}(e|f_1^J, e_1^I) = \frac{p_{lev}(e, f_1^J, e_1^I)}{\sum_{e'} p_{lev}(e', f_1^J, e_1^I)} \tag{6}$$

where

$$p_{lev}(e, f_1^J, e_1^I) = \sum_{n=1}^{N} \delta(e, L_i(e_1^I, e_{n,1}^{n,I_n})) \cdot p(f_1^J, e_{n,1}^{n,I_n}) \tag{7}$$

In Eq. (7), $p(f_1^J, e_{n,1}^{n,I_n})$ is the posterior probability of each hypothesis in the N-best list, which is given by the SMT system. $\delta(x, y)$ is the Kronecker function as in Eq. (4).

3.2 Linguistic Features

3.2.1 Syntactic Features

Xiong et al. extracted syntactical feature by checking whether a word is connected with other words from the output of the LG parser. When the parser fails to parse the entire sentence, it ignores one word each time until it finds linkages for remaining words. After parsing, those ignored words which are not connected to any other words to be called *null*-linked words. These *null*-linked words are prone to be syntactically incorrect and the linked words are prone to be syntactically correct, then a binary syntactic feature for a word according to its links can be defined as in (8),

$$link(e) = \begin{cases} yes & \texttt{e has links with other words} \\ no & \texttt{otherwise} \end{cases} \tag{8}$$

Refer to detailed description in [8].

3.3 Lexical Features

Lexical features such as the word itself and the POS [12] are common features used in NLP tasks. In this paper, we also utilize the word/pos with its context (e.g. the previous two words/pos and next two words/pos) to form a feature vector as follows,

– *word*: $(w_{-2}, w_{-1}, w, w_1, w_2)$
– *pos*: $(pos_{-2}, pos_{-1}, pos, pos_1, pos_2)$

4 Classifiers and Feature Representation

In this paper, the translation error detection is regarded as a classification task. In this section, we introduce two kinds of classifiers – MaxEnt and SVM, and then come up with a multi-classifier combination strategy to perform our translation error detection task.

Our translation error detection is a binary classification task that annotates a word e of the translation hypothesis e_1^I as "*correct*" if it is translated correctly, or "*incorrect*" if it is a wrong translation. Therefore, the label set for the classification task can be denoted as $\boldsymbol{y} = \{c, i\}$, where $\boldsymbol{y}$ indicates the label set, c stands for class "*correct*" and i represents class "*incorrect*".

4.1 Maximum Entropy Classifier

The MaxEnt model is the most commonly-used classifier in NLP tasks, which is a generalization of the model used by the naive Bayes classifier. The basic idea of MaxEnt model is to build a consistent model for all known factors without considering any unknown factors. A remarkable characteristic of the MaxEnt classifier is that features are not necessarily required independent. In doing so, features can be arbitrarily added into the model. Denote the binary classification samples as $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ ($\mathbf{x}_i$ represents the feature vector, $y_i \in \{c, i\}$) (c and i stand for *correct* and *incorrect* labels respectively in our task), then as in literature [8], the MaxEnt classifier for a word e in a hypothesis can be formulated as in (9),

$$p(y|\mathbf{x}) = \frac{exp(\sum_i(\lambda_i f_i(\mathbf{x}, y)))}{\sum_y(exp(\sum_i(\lambda_i f_i(\mathbf{x}, y))))} \tag{9}$$

where f_i is a feature function, λ_i is the weight of f_i, y is the class label, and $\mathbf{x}$ is the feature vector.

4.2 SVM Classifier

SVM has been widely and successfully used in many NLP tasks, such as word sense disambiguation, name entity recognition etc. The basic principle of SVM is to find an optimal hyperplane to make the distance maximum between two classes. The classification task can be defined as in (10),

$$g(x) = sign[\sum_1^n a_i y_i K(x_i, x) + b] \tag{10}$$

where $g(x)$ is the optimal classification hyperplane, $K(x_i, x)$ is the kernel function.

4.3 Multi-classifier Combination

Multiple classifiers combination method has been applied in many NLP tasks, such as word sense disambiguation etc. Most of these applications have shown a considerable improvement over the performance of individual classifiers. Therefore, it leads us to consider implementing such a multiple classifier combination strategy for the translation error detection task as well.

In general, different types of classifiers would reflect different characteristics in the classification results, so that using classifier combination techniques can potentially achieve a better classification accuracy based on the assumption that the errors made by each of the classifiers are not identical, and if the combination strategy is appropriate, then the outcome might correct some errors.

Several effective ways of classifier combination techniques have been studied, such as probability distribution based method, vote-based method, rank-based method, linear/weighted linear combination method etc. [13–15]. Regarding this task, considering that we only have two different classifiers, a straightforward strategy – probability product rule – is presented to combine the outputs of two individual classifiers to achieve better results.

The algorithm of the probability product for our translation error detection task can be formalized as:

Assume the task is a binary classification, given the classifier set $C = \{C_1, \ldots, C_n\}$ and the class set $c = \{c_1, c_2\}$, for a word sequence $S = \{w_1, \ldots, w_m\}$ in which each word w_i need to be tagged as c_1 or c_2, if the outputs for a word w from each individual classifier C_i can be denoted as $O_w^i = \{p_{c_1}^i, p_{c_2}^i\}$, in which $p^i_{c_1}$ indicates the probability that the word w is tagged as c_1 by the classifier C_i, and $p^i_{c_2}$ is the probability that w is tagged as c_2 by the classifier C_i, conditioned by $p^i_{c_1} + p^i_{c_2} = 1$, then the probability product algorithm for the classifier combination can be formulated as in (11),

$$c_w = \max\{\prod_{i=1}^{n} p^i_{c_1}, \prod_{i=1}^{n} p^i_{c_2}\} \tag{11}$$

where c_w indicates the predicted class for the word w. In our task, $n = 2$.

4.4 Feature Vector Representation

As described in previous sections, in our translation error detection task, we have four kinds of features: *wpp*, *pos*, *word* and *link* (c.f. Section 3). In this section, we introduce how to construct a normalized and unified feature vector format for the MaxEnt and SVM classifiers.

Generally in the NLP classification task, context information is usually to be considered in the process of feature extraction. Therefore, in our task, to build a feature vector for a word e, we look at 2 words before and 2 words after the current word position as well. Thus, the feature vector $\mathbf{x}$ that includes four kinds of features can be denoted as,

$$\begin{aligned}\mathbf{x} = <\ & wpp_{-2}, wpp_{-1}, wpp, wpp_1, wpp_2, pos_{-2}, pos_{-1},\\ & pos, pos_1, pos_2, word_{-2}, word_{-1}, word, word_1,\\ & word_2, link_{-2}, link_{-1}, link, link_1, link_2 >\end{aligned}$$

As to the individual classifiers, we use the MaxEnt toolkit [3] as our MaxEnt classifier, and use LibSVM [4] as our SVM classifier [16] respectively.

5 Experiments and Analysis

5.1 Chinese-English SMT System

We utilize Moses [17] to provide 10,000-best list with translation direction from Chinese to English. The training data consists of 3,397,538 pairs of sentences (including Hong Kong news, FBIS, ISI Chinese-English Network Data and Xin-Hua news etc.). The language model is five-gram built on the English part of the bilingual corpus and Xinhua part of the English Gigaword.

The development set for SMT training is the current set of NIST MT 2006 (1,664 source sentences) and the test sets are NIST MT-05 (1,082 sentences) and NIST MT-08 (1,357 sentences). Each source sentence has four references. During the decoding process, the SMT system exports 10,000-best hypotheses for each source sentence, i.e., $N = 10,000$.

Performance of SMT systems on two test sets is shown in Table 1 in terms of BLEU4 and other metrics.

[3] `http://homepages.inf.ed.ac.uk/s0450736/maxenttoolkit.html.`
[4] Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

Table 1. SMT performance and the ratio of correct words (RCW)

dataset	BLEU4(%)	WER(%)	TER(%)	RCW(%)
NIST MT 2008	25.97	69.79	63.56	37.99
NIST MT 2005	33.17	69.50	61.40	41.59

5.2 Experimental Settings for Translation Error Detection Task

Development and test sets: in the error detection task, we use NIST MT-08 as the development set to tune the classifiers, and NIST MT-05 as the test set to evaluate the classification performance.
Data annotation: we use the WER metric in TER toolkit [18] to determine the true labels for the words in the development set and the test set. Specifically, we firstly perform the minimum edit distance alignment between the hypothesis and the four references, and then select the one with minimum WER score as the final reference to tag the hypothesis. That is, a word e in the hypothesis is tagged as c if it is the same as that in the reference, otherwise tag it as i.

There are 14,658 correct words and 23,929 incorrect words in the 1-best hypothesis of MT-08 set (37.99% ratio of correct words, RCW), 15,179 correct words and 21,318 incorrect words in the 1-best hypothesis of MT-05 set (41.59% RCW). See RCW in Table 1.

Evaluation Metrics: the commonly-used evaluation metrics for the classification task includes CER (classification error rate), precision, recall and F measure. In our translation error detection task, we use CER as the main evaluation metric to evaluate the system performance that is defined as in (12),

$$\texttt{CER} = \frac{\#\texttt{of wrongly tagged words}}{\#\texttt{of total words}} \tag{12}$$

Since the RCW is less than 50% (41.59%), i.e., the number of incorrect words is more than correct words, it is reasonable to use the RCW as the baseline of CER to examine the classification performance of classifiers.

We also use F measure, Precision and Recall as the auxiliary evaluation metrics to evaluate some performance of features and classifiers. See definitions in [8].

5.3 Classification Experiments for Individual Classifiers

Results of different features and feature combinations over two different individual classifiers are shown in Table 2.

WPP_Dir represents the fixed position-based WPP, *WPP_Win* represents the flexible position-based WPP with the window size 2, and *WPP_Lev* represents word alignment-based WPP. *com*1 represents the feature combination of *WPP_Dir + Word + Pos + Link*, *com*2 stands for the feature combination of

Table 2. Results of two individual classifiers for translation error detection

	MaxEnt				SVM			
Feature	CER(%)	P(%)	R(%)	F(%)	CER(%)	P(%)	R(%)	F(%)
Baseline	*41.59*	–	–	–	*41.59*	–	–	–
WPP_Dir	*40.48*	63.44	72.46	67.65	***37.64***	61.19	97.20	75.11
WPP_Win	*39.70*	63.82	73.95	68.51	***37.47***	61.31	97.17	75.18
WPP_Lev	*40.12*	60.24	92.07	72.83	***37.37***	61.37	97.24	75.25
word	*39.11*	64.20	76.67	69.04	***37.68***	64.06	80.84	71.48
pos	*39.50*	61.52	86.46	71.89	***39.12***	62.10	84.75	73.68
link	*40.89*	59.55	93.55	72.77	***37.70***	61.36	95.71	74.78
com1	*35.93*	63.93	88.30	74.17	***35.36***	63.93	90.57	74.95
com2	*35.55*	64.77	85.83	73.83	***35.27***	64.11	89.98	74.86
com3	*35.62*	65.31	83.22	73.15	***35.44***	64.02	89.81	74.75

WPP_Win + Word + Pos + Link, and *com*3 indicates the feature combination of *WPP_Lev+ Word + Pos + Link*.

We can see that 1) all these individual features over two classifiers significantly reduce the CER compared to the baseline; 2) the *WPP_Win* and *WPP_Lev* perform better than *WPP_Dir* which shows that position information is helpful; 3) linguistic features perform better than three WPP features over the MaxEnt (except *link*), while worse than those over the SVM classifier in terms of CER. However, they all significantly reduce the CER compared to the baseline; 4) *WPP_Win + word + pos + link* obtains the best performance both on MaxEnt and SVM classifiers. Feature combinations outperform any of the individual features; 5) SVM classifier outperforms the MaxEnt classifier on all features in terms of the CER.

5.4 Classification Experiment on Multi-classifier Combination Strategy

The results of the Multi-classifier Combination experiment are shown in Table 3.

Table 3. Results of multi-classifier combination for translation error detection

	MaxEnt		SVM		Multi-classifier	
Feature	CER(%)	F(%)	CER(%)	F(%)	CER(%)	F(%)
Baseline	*41.59*	–	*41.59*	–	*41.59*	–
com1	*35.93*	74.17	*35.36*	74.95	***35.31***	74.55
com2	*35.55*	73.83	*35.27*	74.86	***34.94***	74.55
com3	*35.62*	73.15	*35.44*	74.75	***34.90***	74.47

We can see from the results that 1) compared to the baseline, the proposed multi-classifier combination method achieved significant improvement by relative 15.10%, 15.99% and 16.09% in terms of CER. 2) compared to the MaxEnt and SVM classifiers over three feature combinations, namely *com*1, *com*2 and *com*3, the proposed multi-classifier combination method achieved significant improvement respectively by relative 0.15%, 0.94%, 1.52%, and 1.73%, 1.72%, 2.02% in terms of CER. 3) *WPP_Lev + word + pos + link* and *WPP_Win + word + pos + link* are significantly better than *WPP_Dir + word + pos + link*, which indicates that the flexible position based WPP feature is more useful than the fixed position based WPP on the multi-classifier combination.

From the comparison of the results, we can conclude: 1) generally speaking, *WPP_Win* performs the best and robust both in the three individual WPP features and the three combined features. The reason we consider is that the sliding window makes the alignment more flexible and considers more context information. 2) linguistic features are helpful to the error detection. 3) multi-classifier strategy is effective to further improve the error detection performance.

Based on the observations, we also found that 1) the name entities (person name, location name, organization name etc.) are prone to be wrongly classified; 2) the prepositions, conjunctions, auxiliary verbs and articles are easier to be wrongly classified due to the factors that they often have an impact on the word orders or lead to empty alignment links; 3) the proportion of the notional words that are wrongly classified is relatively small.

Conclusions and Future Work

This paper presents a multi-classifier combination strategy for translation error detection. Firstly three different kinds of WPP features, three linguistic features and two individual classifiers are introduced, then a probability product based multi-classifier combination method is proposed which multiplies the corresponding classification probabilities respectively coming from MaxEnt and SVM classifiers for each word in a hypothesis, and then decides the label by the maximum probability. Experimental results on Chinese-to-English NIST MT data sets show that 1) in terms of individual classifiers used in our experiments, SVM classifier outperforms the MaxEnt classifier; 2) the proposed multi-classifier combination method performs the best compared to two individual classifiers.

In future work, we intend to carry out further study on the error detection task in the respects of 1) introducing paraphrases to annotate the hypotheses so that it can truly reflect the *correct* or *incorrect* at the semantic level; 2) introducing new useful features to further improve the detection capability; 3) performing experiments on more language pairs to verify our proposed method.

Acknowledgments

This work is supported by NSF project (61100085), SRF for ROCS, State Education Ministry, and Research Foundation of Education Department of Shaanxi

Provincial Government (11JK1029). Thanks the reviewers for their insightful comments and suggestions.

References

[1] DeCamp, J.: What is missing in user-centric MT? In Proceedings of MT Summit XII, pages 489–495 (2009)
[2] Roturier, J.: Deploying novel MT technology to raise the bar for quality: a review of key advantages and challenges. In Proceedings of MT Summit XII, pages 1–8 (2009)
[3] Simard, M. and Isabelle, P.: Phrase-based machine translation in a computer-assisted translation environment. In Proceedings of MT Summit XII, pages 120–127 (2009)
[4] Gandrabur, S. and Foster, G.: Confidence Estimation for Translation Predicion. In Proceedings of the HLT-NAACL, pages 95–102 (2003)
[5] Ueffing, N., Macherey, K., and Ney, H.: Confidence Measures for Statistical Machine Translation. In Proceedings of MT Summit IX, pages 169–176 (2003)
[6] Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., C. Goutte, A. Kuesza, A. S., and Ueffing,N.: Confidence Estimation for Machine Translation. In Proceedings of COLING, pages 315–321 (2004)
[7] Ueffing, N. and Ney, H.: Word-Level Confidence Estimation for Machine Translation. Computational Linguistics, 33(1):9–40 (2007)
[8] Xiong, D., Zhang, M., and Li, H.: Error detection for statistical machine translation using linguistic features. In Proceedings of ACL, pages 604–611 (2010)
[9] Bach, N., Huang, F., and AI-Onaizan, Y.: Goodness: A Method for Measuring Machine Translation Confidence. In Proceedings of ACL, pages 211–219 (2011)
[10] Specia, L., Cancedda, N., Dymetman, M., MarcoTurchi, and Cristianini, N.: Estimating the sentence-level quality of machine translation systems. In Proceedings of EAMT, pages 28–35 (2009)
[11] Specia, L., Saunders, C., Turchi, M., Wang, Z., and Shawe-Taylor, J.: Improving the confidence of machine translation quality estimates. In Proceedings of MT Summit, pages 136–143 (2009)
[12] Ratnaparkhi, A.: A Maximum Entropy Model for Part-Of-Speech Tagging. In Proceedings of the Empirical Methods in Natural Language Processing Conference (EMNLP), pages 133–142, Philadelphia (1996)
[13] Battiti, R. and Colla, A.: Democracy in Neural Nets: Voting Schemes for Classification. Neural Networks, 7(4):691–707 (1994)
[14] Kittler, J., Hatef, M., Duin, R., and Matas, J.: On combining classifiers. IEEE Transaction on Pattern Analysis and Machine Intellifence, 20(3):226–239 (1998)
[15] Florian, R., Cucerzan, S., Schafer, C., and Yarowsky, D.: Combining Classifiers for word sense disambiguation. Natural Language Engineering, 8(4):327–341 (2002).
[16] Chang, C.-C. and Lin, C.-J.: LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 3(2):27:1–27 (2011)
[17] 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., and Herbst, E.: Moses: open source toolkit for statistical machine translation. In Proceedings of ACL, pages 177–180 (2010)
[18] Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J.: A study of translation edit rate with targeted human annotation. In Proceedings of the 7th Conference of the Association for Machine Translation in the Americas, pages 223–231, MD (2006)