Error Detection for Statistical Machine Translation Using Linguistic Features

Deyi Xiong, Min Zhang, Haizhou Li
Human Language Technology
Institute for Infocomm Research
1 Fusionopolis Way, #21-01 Connexis, Singapore 138632.
{dyxiong, mzhang, hli}@i2r.a-star.edu.sg

Abstract

Automatic error detection is desired in the post-processing to improve machine translation quality. The previous work is largely based on confidence estimation using system-based features, such as word posterior probabilities calculated from N-best lists or word lattices. We propose to incorporate two groups of linguistic features, which convey information from outside machine translation systems, into error detection: lexical and syntactic features. We use a maximum entropy classifier to predict translation errors by integrating word posterior probability feature and linguistic features. The experimental results show that 1) linguistic features alone outperform word posterior probability based confidence estimation in error detection; and 2) linguistic features can further provide complementary information when combined with word confidence scores, which collectively reduce the classification error rate by 18.52% and improve the F measure by 16.37%.

1 Introduction

Translation hypotheses generated by a statistical machine translation (SMT) system always contain both correct parts (e.g. words, n-grams, phrases matched with reference translations) and incorrect parts. Automatically distinguishing incorrect parts from correct parts is therefore very desirable not only for post-editing and interactive machine translation (Ueffing and Ney, 2007) but also for SMT itself: either by rescoring hypotheses in the N-best list using the probability of correctness calculated for each hypothesis (Zens and Ney, 2006) or by generating new hypotheses using N-best lists from one SMT system or multiple systems (Akibay et al., 2004; Jayaraman and Lavie, 2005).

In this paper we restrict the "parts" to words. That is, we detect errors at the word level for SMT. A common approach to SMT error detection at the word level is calculating the confidence at which a word is correct. The majority of word confidence estimation methods follows three steps:

1) Calculate features that express the correctness of words either based on SMT model (e.g. translation/language model) or based on SMT system output (e.g. N-best lists, word lattices) (Blatz et al., 2003; Ueffing and Ney, 2007).

2) Combine these features together with a classification model such as multi-layer perceptron (Blatz et al., 2003), Naive Bayes (Blatz et al., 2003; Sanchis et al., 2007), or log-linear model (Ueffing and Ney, 2007).

3) Divide words into two groups (correct translations and errors) by using a classification threshold optimized on a development set.

Sometimes the step 2) is not necessary if only one effective feature is used (Ueffing and Ney, 2007); and sometimes the step 2) and 3) can be merged into a single step if we directly output predicting results from binary classifiers instead of making thresholding decision.

Various features from different SMT models and system outputs are investigated (Blatz et al., 2003; Ueffing and Ney, 2007; Sanchis et al., 2007; Raybaud et al., 2009). Experimental results show that they are useful for error detection. However, it is not adequate to just use these features as discussed in (Shi and Zhou, 2005) because the information that they carry is either from the inner components of SMT systems or from system outputs. To some extent, it has already been considered by SMT systems. Hence finding external information sources from outside SMT systems is desired for error detection.

Linguistic knowledge is exactly such a good choice as an external information source. It has already been proven effective in error detection for speech recognition (Shi and Zhou, 2005). However, it is not widely used in SMT error detection. The reason is probably that people have yet to find effective linguistic features that outperform non-linguistic features such as word posterior probability features (Blatz et al., 2003; Raybaud et al., 2009). In this paper, we would like to show an effective use of linguistic features in SMT error detection.

We integrate two sets of linguistic features into a maximum entropy (MaxEnt) model and develop a MaxEnt-based binary classifier to predict the category (correct or incorrect) for each word in a generated target sentence. Our experimental results show that linguistic features substantially improve error detection and even outperform word posterior probability features. Further, they can produce additional improvements when combined with word posterior probability features.

The rest of the paper is organized as follows. In Section 2, we review the previous work on word-level confidence estimation which is used for error detection. In Section 3, we introduce our linguistic features as well as the word posterior probability feature. In Section 4, we elaborate our MaxEnt-based error detection model which combine linguistic features and word posterior probability feature together. In Section 5, we describe the SMT system which we use to generate translation hypotheses. We report our experimental results in Section 6 and conclude in Section 7.

2 Related Work

In this section, we present an overview of confidence estimation (CE) for machine translation at the word level. As we are only interested in error detection, we focus on work that uses confidence estimation approaches to detect translation errors. Of course, confidence estimation is not limited to the application of error detection, it can also be used in other scenarios, such as translation prediction in an interactive environment (Grandrabur and Foster, 2003) .

In a JHU workshop, Blatz et al. (2003) investigate using neural networks and a naive Bayes classifier to combine various confidence features for confidence estimation at the word level as well as at the sentence level. The features they use for word level CE include word posterior probabilities estimated from N-best lists, features based on SMT models, semantic features extracted from WordNet as well as simple syntactic features, i.e. parentheses and quotation mark check. Among all these features, the word posterior probability is the most effective feature, which is much better than linguistic features such as semantic features, according to their final results.

Ueffing and Ney (2007) exhaustively explore various word-level confidence measures to label each word in a generated translation hypothesis as correct or incorrect. All their measures are based on word posterior probabilities, which are estimated from 1) system output, such as word lattices or N-best lists and 2) word or phrase translation table. Their experimental results show that word posterior probabilities directly estimated from phrase translation table are better than those from system output except for the Chinese-English language pair.

Sanchis et al. (2007) adopt a smoothed naive Bayes model to combine different word posterior probability based confidence features which are estimated from N-best lists, similar to (Ueffing and Ney, 2007).

Raybaud et al. (2009) study several confidence features based on mutual information between words and n-gram and backward n-gram language model for word-level and sentence-level CE. They also explore linguistic features using information from syntactic category, tense, gender and so on. Unfortunately, such linguistic features neither improve performance at the word level nor at the sentence level.

Our work departs from the previous work in two major respects.

- We exploit various linguistic features and show that they are able to produce larger improvements than widely used system-related features such as word posterior probabilities. This is in contrast to some previous work. Yet another advantage of using linguistic features is that they are system-independent, which therefore can be used across different systems.

- We treat error detection as a complete binary classification problem. Hence we directly output prediction results from our discriminatively trained classifier without optimizing a classification threshold on a distinct development set beforehand.[1] Most previous approaches make decisions based on a pre-tuned classification threshold τ as follows

$$class = \begin{cases} correct, & \Phi(correct, \theta) > \tau \\ incorrect, & otherwise \end{cases}$$

where Φ is a classifier or a confidence measure and θ is the parameter set of Φ. The performance of these approaches is strongly dependent on the classification threshold.

3 Features

We explore two sets of linguistic features for each word in a machine generated translation hypothesis. The first set of linguistic features are simple lexical features. The second set of linguistic features are syntactic features which are extracted from link grammar parse. To compare with the previously widely used features, we also investigate features based on word posterior probabilities.

3.1 Lexical Features

We use the following lexical features.

- wd: word itself
- pos: part-of-speech tag from a tagger trained on WSJ corpus. [2]

For each word, we look at previous n words/tags and next n words/tags. They together form a word/tag sequence pattern. The basic idea of using these features is that words in rare patterns are more likely to be incorrect than words in frequently occurring patterns. To some extent, these two features have similar function to a target language model or pos-based target language model.

3.2 Syntactic Features

High-level linguistic knowledge such as syntactic information about a word is a very natural and promising indicator to decide whether this word is syntactically correct or not. Words occurring in an ungrammatical part of a target sentence are prone to be incorrect. The challenge of using syntactic knowledge for error detection is that machine-generated hypotheses are rarely fully grammatical. They are mixed with grammatical and ungrammatical parts, which hence are not friendly to traditional parsers trained on grammatical sentences because ungrammatical parts of a machine-generated sentence could lead to a parsing failure.

To overcome this challenge, we select the *Link Grammar* (LG) parser [3] as our syntactic parser to generate syntactic features. The LG parser produces a set of labeled links which connect pairs of words with a link grammar (Sleator and Temperley, 1993).

The main reason why we choose the LG parser is that it provides a robustness feature: *null-link* scheme. The null-link scheme allows the parser to parse a sentence even when the parser can not fully interpret the entire sentence (e.g. including ungrammatical parts). When the parser fail to parse the entire sentence, it ignores one word each time until it finds linkages for remaining words. After parsing, those ignored words are not connected to any other words. We call them *null-linked* words.

Our hypothesis is that null-linked words are prone to be syntactically incorrect. We hence straightforwardly define a syntactic feature for a word w according to its links as follows

$$link(w) = \begin{cases} yes, & w \ has \ links \\ no, & otherwise \end{cases}$$

In Figure 1 we show an example of a generated translation hypothesis with its link parse. Here links are denoted with dotted lines which are annotated with link types (e.g., Jp, Op). Bracketed words, namely "," and "including", are null-linked words.

3.3 Word Posterior Probability Features

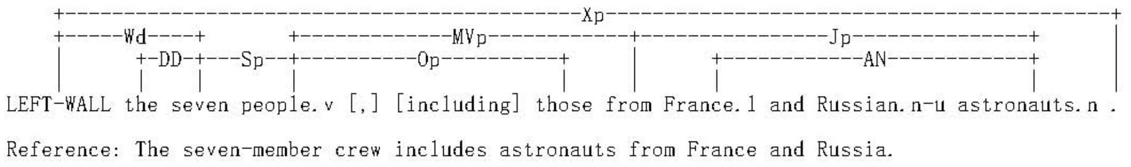
Our word posterior probability is calculated on N-best list, which is first proposed by (Ueffing et al., 2003) and widely used in (Blatz et al., 2003; Ueffing and Ney, 2007; Sanchis et al., 2007).

Given a source sentence f, let $\{e_n\}_1^N$ be the N-best list generated by an SMT system, and let e_n^i is the i-th word in e_n. The major work of calculating word posterior probabilities is to find the Levenshtein alignment (Levenshtein, 1966) between the best hypothesis e_1 and its competing hypothesis

[1] This does not mean we do not need a development set. We do validate our feature selection and other experimental settings on the development set.

[2] Available via http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/postagger/

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



```
    +-------------------------------------------Xp--------------------------------------------+
    +--Wd--+            +------------MVp-------------+--------------Jp----------------+       |
    |      +-DD-+--Sp---+---------Op-----------+     |                    +----AN-----+       |
    |      |    |       |                      |     |                    |           |       |
LEFT-WALL the seven people.v [,] [including] those from France.l and Russian.n-u astronauts.n .

Reference: The seven-member crew includes astronauts from France and Russia.
```

Figure 1: An example of Link Grammar parsing results.

e_n in the N-best list $\{e_n\}_1^N$. We denote the alignment between them as $\ell(e_1, e_n)$. The word in the hypothesis e_n which e_1^i is Levenshtein aligned to is denoted as $\ell_i(e_1, e_n)$.

The word posterior probability of e_1^i is then calculated by summing up the probabilities over all hypotheses containing e_1^i in a position which is Levenshtein aligned to e_1^i.

$$p_{wpp}(e_1^i) = \frac{\sum_{e_n:\,\ell_i(e_1,e_n)=e_1^i} p(e_n)}{\sum_1^N p(e_n)}$$

To use the word posterior probability in our error detection model, we need to make it discrete. We introduce a feature for a word w based on its word posterior probability as follows

$$dwpp(w) = \lfloor -log(p_{wpp}(w))/df \rfloor$$

where df is the discrete factor which can be set to 1, 0.1, 0.01 and so on. "$\lfloor\ \rfloor$" is a rounding operator which takes the largest integer that does not exceed $-log(p_{wpp}(w))/df$. We optimize the discrete factor on our development set and find the optimal value is 1. Therefore a feature "$dwpp =$ 2" represents that the logarithm of the word posterior probability is between -3 and -2;

4 Error Detection with a Maximum Entropy Model

As mentioned before, we consider error detection as a binary classification task. To formalize this task, we use a feature vector ψ to represent a word w in question, and a binary variable c to indicate whether this word is correct or not. In the feature vector, we look at 2 words before and 2 words after the current word position $(w_{-2}, w_{-1}, w, w_1, w_2)$. We collect features $\{wd, pos, link, dwpp\}$ for each word among these words and combine them into the feature vector ψ for w. As such, we want the feature vector to capture the contextual environment, e.g., *pos* sequence pattern, syntactic pattern, where the word w occurs.

For classification, we employ the maximum entropy model (Berger et al., 1996) to predict whether a word w is correct or incorrect given its feature vector ψ.

$$p(c|\psi) = \frac{exp(\sum_i \theta_i f_i(c, \psi))}{\sum_{c'} exp(\sum_i \theta_i f_i(c', \psi))}$$

where f_i is a binary model feature defined on c and the feature vector ψ. θ_i is the weight of f_i. Table 1 shows some examples of our binary model features.

In order to learn the model feature weights θ for probability estimation, we need a training set of m samples $\{\psi^i, c^i\}_1^m$. The challenge of collecting training instances is that the correctness of a word in a generated translation hypothesis is not intuitively clear (Ueffing and Ney, 2007). We will describe the method to determine the correctness of a word in Section 6.1, which is broadly adopted in previous work.

We tune our model feature weights using an off-the-shelf MaxEnt toolkit (Zhang, 2004). To avoid overfitting, we optimize the Gaussian prior on the development set. During test, if the probability $p(correct|\psi)$ is larger than $p(incorrect|\psi)$ according the trained MaxEnt model, the word is labeled as correct otherwise incorrect.

5 SMT System

To obtain machine-generated translation hypotheses for our error detection, we use a state-of-the-art phrase-based machine translation system MOSES (Koehn et al., 2003; Koehn et al., 2007). The translation task is on the official NIST Chinese-to-English evaluation data. The training data consists of more than 4 million pairs of sentences (including 101.93M Chinese words and 112.78M English words) from LDC distributed corpora. Table 2 shows the corpora that we use for the translation task.

We build a four-gram language model using the SRILM toolkit (Stolcke, 2002), which is trained

Feature	Example
wd	$f(c,\psi) = \begin{cases} 1, & \psi.w.wd = ".", c = correct \\ 0, & otherwise \end{cases}$
pos	$f(c,\psi) = \begin{cases} 1, & \psi.w_2.pos = "NN", c = incorrect \\ 0, & otherwise \end{cases}$
$link$	$f(c,\psi) = \begin{cases} 1, & \psi.w.link = no, c = incorrect \\ 0, & otherwise \end{cases}$
$dwpp$	$f(c,\psi) = \begin{cases} 1, & \psi.w_{-2}.dwpp = 2, c = correct \\ 0, & otherwise \end{cases}$

Table 1: Examples of model features.

LDC ID	Description
LDC2004E12	United Nations
LDC2004T08	Hong Kong News
LDC2005T10	Sinorama Magazine
LDC2003E14	FBIS
LDC2002E18	Xinhua News V1 beta
LDC2005T06	Chinese News Translation
LDC2003E07	Chinese Treebank
LDC2004T07	Multiple Translation Chinese

Table 2: Training corpora for the translation task.

on Xinhua section of the English Gigaword corpus (181.1M words). For minimum error rate tuning (Och, 2003), we use NIST MT-02 as the development set for the translation task. In order to calculate word posterior probabilities, we generate 10,000 best lists for NIST MT-02/03/05 respectively. The performance, in terms of BLEU (Papineni et al., 2002) score, is shown in Table 4.

6 Experiments

We conducted our experiments at several levels. Starting with MaxEnt models with single linguistic feature or word posterior probability based feature, we incorporated additional features incrementally by combining features together. In doing so, we would like the experimental results not only to display the effectiveness of linguistic features for error detection but also to identify the additional contribution of each feature to the task.

6.1 Data Corpus

For the error detection task, we use the best translation hypotheses of NIST MT-02/05/03 generated by MOSES as our training, development, and test corpus respectively. The statistics about these corpora is shown in Table 3. Each translation hypothesis has four reference translations.

	Corpus	Sentences	Words
Training	MT-02	878	24,225
Development	MT-05	1082	31,321
Test	MT-03	919	25,619

Table 3: Corpus statistics (number of sentences and words) for the error detection task.

To obtain the linkage information, we run the LG parser on all translation hypotheses. We find that the LG parser can not fully parse 560 sentences (63.8%) in the training set (MT-02), 731 sentences (67.6%) in the development set (MT-05) and 660 sentences (71.8%) in the test set (MT-03). For these sentences, the LG parser will use the the null-link scheme to generate null-linked words.

To determine the true class of a word in a generated translation hypothesis, we follow (Blatz et al., 2003) to use the word error rate (**WER**). We tag a word as correct if it is aligned to itself in the Levenshtein alignment between the hypothesis and the nearest reference translation that has minimum edit distance to the hypothesis among four reference translations. Figure 2 shows the Levenshtein alignment between a machine-generated hypothesis and its nearest reference translation. The "Class" row shows the label of each word according to the alignment, where "c" and "i" represent *correct* and *incorrect* respectively.

There are several other metrics to tag single words in a translation hypothesis as correct or incorrect, such as **PER** where a word is tagged as correct if it occurs in one of reference translations with the same number of occurrences, **Set** which is a less strict variant of PER, ignoring the number of occurrences per word. In Figure 2, the two words "last year" in the hypothesis will be tagged as correct if we use the PER or Set metric since they do not consider the occurring positions of words. Our

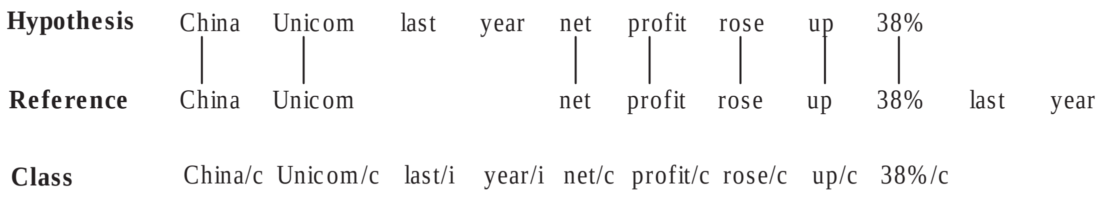

Hypothesis
China Unicom last year net profit rose up 38%
Reference
China Unicom net profit rose up 38% last year
Class
China/c Unicom/c last/i year/i net/c profit/c rose/c up/c 38%/c

Figure 2: Tagging a word as correct/incorrect according to the Levenshtein alignment.

Corpus	BLEU (%)	RCW (%)
MT-02	33.24	47.76
MT-05	32.03	47.85
MT-03	32.86	47.57

Table 4: Case-insensitive BLEU score and ratio of correct words (RCW) on the training, development and test corpus.

metric corresponds to the **m-WER** used in (Ueffing and Ney, 2007), which is stricter than PER and Set. It is also stricter than normal WER metric which compares each hypothesis to all references, rather than the nearest reference.

Table 4 shows the case-insensitive BLEU score and the percentage of words that are labeled as correct according to the method described above on the training, development and test corpus.

6.2 Evaluation Metrics

To evaluate the overall performance of the error detection, we use the commonly used metric, classification error rate (CER) to evaluate our classifiers. CER is defined as the percentage of words that are wrongly tagged as follows

$$CER = \frac{\text{\# of wrongly tagged words}}{\text{Total \# of words}}$$

The baseline CER is determined by assuming the most frequent class for all words. Since the ratio of correct words in both the development and test set is lower than 50%, the most frequent class is "incorrect". Hence the baseline CER in our experiments is equal to the ratio of correct words as these words are wrongly tagged as incorrect.

We also use precision and recall on errors to evaluate the performance of error detection. Let n_g be the number of words of which the true class is incorrect, n_t be the number of words which are tagged as incorrect by classifiers, and n_m be the number of words tagged as incorrect that are indeed translation errors. The precision Pre is the percentage of words correctly tagged as translation errors.

$$Pre = \frac{n_m}{n_t}$$

The recall Rec is the proportion of actual translation errors that are found by classifiers.

$$Rec = \frac{n_m}{n_g}$$

F measure, the trade-off between precision and recall, is also used.

$$F = \frac{2 \times Pre \times Rec}{Pre + Rec}$$

6.3 Experimental Results

Table 5 shows the performance of our experiments on the error detection task. To compare with previous work using word posterior probabilities for confidence estimation, we carried out experiments using wpp estimated from N-best lists with the classification threshold τ, which was optimized on our development set to minimize CER. A relative improvement of 9.27% is achieved over the baseline CER, which reconfirms the effectiveness of word posterior probabilities for error detection.

We conducted three groups of experiments using the MaxEnt based error detection model with various feature combinations.

- The first group of experiments uses single feature, such as $dwpp$, pos. We find the most effective feature is pos, which achieves a 16.12% relative improvement over the baseline CER and 7.55% relative improvement over the CER of word posterior probability thresholding. Using discrete word posterior probabilities as features in the MaxEnt based error detection model is marginally better than word posterior probability thresholding in terms of CER, but obtains a 13.79% relative improvement in F measure. The syntactic feature $link$ also improves the error detection in terms of CER and particularly recall.

Combination	Features	CER (%)	Pre (%)	Rec (%)	F (%)
Baseline	-	47.57	-	-	-
Thresholding wpp	-	43.16	58.98	58.07	58.52
MaxEnt ($dwpp$)	44	43.07	56.12	81.86	66.59
MaxEnt (wd)	19,164	41.57	58.25	73.11	64.84
MaxEnt (pos)	199	39.90	58.88	79.23	67.55
MaxEnt ($link$)	19	44.31	54.72	89.72	67.98
MaxEnt ($wd + pos$)	19,363	39.43	59.36	78.60	67.64
MaxEnt ($wd + pos + link$)	19,382	39.79	58.74	80.97	68.08
MaxEnt ($dwpp + wd$)	19,208	41.04	57.18	83.75	67.96
MaxEnt ($dwpp + wd + pos$)	19,407	38.88	59.87	78.38	67.88
MaxEnt ($dwpp + wd + pos + link$)	19,426	38.76	59.89	78.94	68.10

Table 5: Performance of the error detection task.

- The second group of experiments concerns with the combination of linguistic features without word posterior probability feature. The combination of lexical features improves both CER and precision over single lexical feature (wd, pos). The addition of syntactic feature $link$ marginally undermines CER but improves recall by a lot.

- The last group of experiments concerns about the additional contribution of linguistic features to error detection with word posterior probability. We added linguistic features incrementally into the feature pool. The best performance was achieved by using all features, which has a relative of improvement of 18.52% over the baseline CER.

The first two groups of experiments show that linguistic features, individually (except for $link$) or by combination, are able to produce much better performance than word posterior probability features in both CER and F measure. The best combination of linguistic features achieves a relative improvement of 8.64% and 15.58% in CER and F measure respectively over word posterior probability thresholding.

The Table 5 also reveals how linguistic features improve error detection. The lexical features (pos, wd) improve precision when they are used. This suggests that lexical features can help the system find errors more accurately. Syntactic features ($link$), on the other hand, improve recall whenever they are used, which indicates that they can help the system find more errors.

We also show the number of features in each combination in Table 5. Except for the wd feature, the pos has the largest number of features, 199, which is a small set of features. This suggests that our error detection model can be learned from a rather small training set.

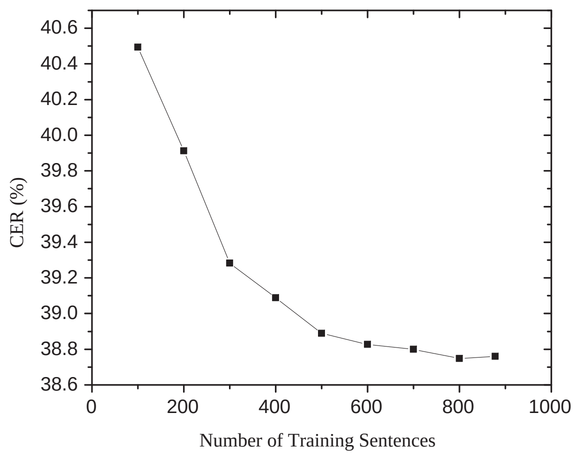


Figure 3: CER vs. the number of training sentences.

Figure 3 shows CERs for the feature combination MaxEnt ($dwpp + wd + pos + link$) when the number of training sentences is enlarged incrementally. CERs drop significantly when the number of training sentences is increased from 100 to 500. After 500 sentences are used, CERs change marginally and tend to converge.

7 Conclusions and Future Work

In this paper, we have presented a maximum entropy based approach to automatically detect errors in translation hypotheses generated by SMT systems. We incorporate two sets of linguistic features together with word posterior probability based features into error detection.

Our experiments validate that linguistic features are very useful for error detection: 1) they by themselves achieve a higher improvement in terms of both CER and F measure than word posterior probability features; 2) the performance is further improved when they are combined with word posterior probability features.

The extracted linguistic features are quite compact, which can be learned from a small training set. Furthermore, The learned linguistic features are system-independent. Therefore our approach can be used for other machine translation systems, such as rule-based or example-based system, which generally do not produce N-best lists.

Future work in this direction involve detecting particular error types such as incorrect positions, inappropriate/unnecessary words (Elliott, 2006) and automatically correcting errors.

References

Yasuhiro Akibay, Eiichiro Sumitay, Hiromi Nakaiway, Seiichi Yamamotoy, and Hiroshi G. Okunoz. 2004. Using a Mixture of N-best Lists from Multiple MT Systems in Rank-sum-based Confidence Measure for MT Outputs. In *Proceedings of COLING.*

Adam L. Berger, Stephen A. Della Pietra andVincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics, 22(1): 39-71.*

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, Nicola Ueffing. 2003. Confidence estimation for machine translation. final report, jhu/clsp summer workshop.

Debra Elliott. 2006 Corpus-based Machine Translation Evaluation via Automated Error Detection in Output Texts. Phd Thesis, University of Leeds.

Simona Gandrabur and George Foster. 2003. Confidence Estimation for Translation Prediction. In *Proceedings of HLT-NAACL.*

S. Jayaraman and A. Lavie. 2005. Multi-engine Machine Translation Guided by Explicit Word Matching. In *Proceedings of EAMT.*

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of HLT-NAACL.*

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constrantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL, Demonstration Session.*

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady, Feb.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL 2003.*

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a Method for Automatically Evaluation of Machine Translation. In *Proceedings of ACL 2002.*

Sylvain Raybaud, Caroline Lavecchia, David Langlois, Kamel Smaïli. 2009. Word- and Sentence-level Confidence Measures for Machine Translation. In *Proceedings of EAMT 2009.*

Alberto Sanchis, Alfons Juan and Enrique Vidal. 2007. Estimation of Confidence Measures for Machine Translation. In *Procedings of Machine Translation Summit XI.*

Daniel Sleator and Davy Temperley. 1993. Parsing English with a Link Grammar. In *Proceedings of Third International Workshop on Parsing Technologies.*

Yongmei Shi and Lina Zhou. 2005. Error Detection Using Linguistic Features. In *Proceedings of HLT/EMNLP 2005.*

Andreas Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proceedings of International Conference on Spoken Language Processing*, volume 2, pages 901-904.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. 2003. Confidence Measures for Statistical Machine Translation. In *Proceedings. of MT Summit IX.*

Nicola Ueffing and Hermann Ney. 2007. Word-Level Confidence Estimation for Machine Translation. *Computational Linguistics, 33(1):9-40.*

Richard Zens and Hermann Ney. 2006. N-gram Posterior Probabilities for Statistical Machine Translation. In *HLT/NAACL: Proceedings of the Workshop on Statistical Machine Translation.*

Le Zhang. 2004. Maximum Entropy Modeling Tooklkit for Python and C++. Available at http://homepages.inf.ed.ac.uk/s0450736 /maxent_toolkit.html.