Mach Translat (2011) 25:1–34
DOI 10.1007/s10590-011-9094-9

"This sentence is wrong." Detecting errors in machine-translated sentences

Sylvain Raybaud · David Langlois · Kamel Smaïli

Received: 18 January 2010 / Accepted: 2 July 2011 / Published online: 20 July 2011

Abstract Machine translation systems are not reliable enough to be used "as is": except for the most simple tasks, they can only be used to grasp the general meaning of a text or assist human translators. The purpose of confidence measures is to detect erroneous words or sentences produced by a machine translation system. In this article, after reviewing the mathematical foundations of confidence estimation, we propose a comparison of several state-of-the-art confidence measures, predictive parameters and classifiers. We also propose two original confidence measures based on Mutual Information and a method for automatically generating data for training and testing classifiers. We applied these techniques to data from the WMT campaign 2008 and found that the best confidence measures yielded an Equal Error Rate of 36.3% at word level and 34.2% at sentence level, but combining different measures reduced these rates to 35.0% and 29.0%, respectively. We also present the results of an experiment aimed at determining how helpful confidence measures are in a post-editing task. Preliminary results suggest that our system is not yet ready to efficiently help post-editors, but we now have both software and a protocol that we can apply to further experiments, and user feedback has indicated aspects which must be improved in order to increase the level of helpfulness of confidence measures.

S. Raybaud (✉) · D. Langlois · K. Smaïli
PAROLE, LORIA, BP 239, 54506 Nancy Cedex, France
e-mail: sylvain.raybaud@loria.fr

D. Langlois
e-mail: david.langlois@loria.fr

K. Smaïli
e-mail: kamel.smaili@loria.fr

Keywords Machine translation · Confidence measure · Translation evaluation · Support vector machine · Mutual information · Partial least squares regression · Logistic regression · Neural network

1 Introduction

A machine translation (MT) system generates the best translation for a given sentence according to a previously learnt or hard-coded model. However no model exists that is able to capture all the subtlety of natural language. Therefore, even the best MT systems make mistakes, and always will; even experts make mistakes after all. Errors take a variety of forms: a word can be wrong, misplaced or missing. Whole translations can be utterly nonsensical or just slightly flawed: involving missing negation, grammatical error and so forth. Therefore, when a document is intended for publication, MT output cannot be used "as is"; at best, it can be used to help a human translator produce good-quality target-language output. A tool for detecting and pinpointing translation errors may ease their work as suggested, for example, in Ueffing and Ney (2005). Gandrabur and Foster (2003) suggest the use of confidence estimation in the context of translation prediction. Confidence estimates could benefit automatic post-editing systems like the one proposed in Simard et al. (2007), by selecting which sentences are to be post-edited. Even end-users using MT for grasping the overall meaning of a text may appreciate the highlighting of dubious words and sentences, thus preventing them from placing too much trust in potentially wrong translations.

However, and maybe because of such high expectations, confidence estimation is a very difficult problem because if decisions are to be made based on these estimations (such as modifying a translation hypothesis), they need to be very accurate in order to maintain translation quality and avoid wasting the user's time. Confidence estimation remains an active research field in numerous domains and much work remains to be done before they can be integrated into working systems.

This article is an overview of many of today's available predictive parameters for MT confidence estimation along with a few original predictive parameters of our own; we also evaluated different machine learning techniques—support vector machines, logistic regression, partial least squares regression and neural networks (Sect. 2)—to combine and optimise them. An exhaustive review would require a whole book, so this paper intends to give a more targeted overview of some of the most significant ideas in the domain. Blatz et al. (2004) proposed a review of many confidence measures for MT. We used this work as a starting point to then carry out a thorough formalisation of the confidence estimation problem and make two contributions to the field:

- Original estimators based on Mutual Information and Part-Of-Speech tags (Sect. 6).
- An algorithm to automatically generate annotated training data for correct/incorrect classifiers (Sect. 4.3).

We implemented techniques described in Siu and Gish (1999) for the evaluation of the performance of the proposed confidence measures. In Sects. 6.2 and 7.2, we show that using a combination of all predictive parameters yields an improvement of

1.3 points absolute in terms of equal error rate over the best parameter used alone (Sect. 3.1).

Finally, we present the results of a post-editing experiment in which we asked volunteers to correct sentences which had been automatically translated and measure their efficiency with and without confidence measures (Sect. 8). Unfortunately, the results suggested that this confidence estimation system was not yet ready to be included in a post-editing software tool. However, we provide a number of useful observations and indications about what is wrong with our system and what is really important for a user.

1.1 Sentence-level confidence estimation

We intuitively recognise a wrong translation that does not have the same meaning as the source sentence, or no meaning at all, or is too disfluent. State-of-the-art natural language processing software is still unable to grasp the meaning of a sentence or to assess its grammatical correctness or fluency, so we have to rely on lower level estimators. The problem is also ill-posed: sometimes one cannot decide what is the meaning of a sentence (especially without a context), let alone decide whether its translation is correct or not (a translation can be correct for one of the possible meanings of the source sentence and wrong for another). In our experiments we asked human judges to assign a numerical score to machine-translated sentences, ranging from one (hopelessly bad translation) to five (perfect) as described in Sect. 4.1. We set the confidence estimation system to automatically detect sentences with scores of three or higher (disfluencies are considered acceptable, insofar as a reader is able to understand the correct meaning in a reasonable amount of time). To this end we computed simple numeric features (also called *predictive parameters*: Language Model (LM) score, length, etc., cf. Sect. 7) and combined them (Sect. 2).

1.2 Word-level confidence estimation

Defining the correctness of a word is even more tricky. Sometimes a translated word may not be appropriate in the context of the source sentence, as may be the case when homonyms are involved (for example if the French word *vol*, speaking of a plane, is translated with the English word *theft* instead of *flight*). In this case the error is obvious but sometimes the correctness of a word might depend on how other words around it are translated. Consider the following example:

$$\text{Ces mots sont presque synonymes} \rightarrow \begin{cases} 1: \text{These words are almost} \\ \quad \text{synonyms } (\textit{correct}) \\ 2: \text{These words have close} \\ \quad \text{meanings } (\textit{correct}) \\ 3: \text{These words have close} \\ \quad \text{synonyms}(\textit{incorrect}) \end{cases}$$

#3 is definitely an incorrect translation but then we have to decide which word is wrong: is it *close*, *synonyms*, *have*, or all of them? In the rest of the article we show

how we trained classifiers to discriminate between correct and incorrect words, but this example shows that no system can ever achieve perfect classification simply because this does not exist.

1.3 Mathematical formulation

Let us now state the problem in mathematically sound terms: the goal of MT is to generate a target sentence from a source sentence. A sentence is a finite sequence of words and punctuation marks, which are elements of the *vocabulary* set. The sentences are represented by random variables. We use the following conventions: a random variable will be represented by a capital letter and a realisation of the variable by the corresponding lower-case letter; bold letters are non-scalar values (sentences, vectors, matrices); non-bold letters are for scalar values like words and real numbers; cursive letters are sets.

$\mathcal{V}_S$	:	Source-language vocabulary
$\mathcal{V}_T$	:	Target-language vocabulary
$\mathbf{S} \in \mathcal{V}_S^*$	:	Sentence in the source language
$\mathbf{T} \in \mathcal{V}_T^*$	:	Sentence in the target language

From these two primary random variables we then derive new variables:

$Len(\mathbf{S}) \in \mathbb{N}$	:	Length of **S** (number of words)
$Len(\mathbf{T}) \in \mathbb{N}$	:	Length of **S**
$S_i \in \mathcal{V}_S$	:	i-th word of **S**
$T_j \in \mathcal{V}_T$	:	j-th word of **T**

When estimating confidence we are given realisations of these variables and then need to guess the values of:

$C_{\mathbf{S},\mathbf{T}} \in \{0, 1\}$	:	correctness of a sentence **T** as a translation of **S**
$C_{\mathbf{S},\mathbf{T},j} \in \{0, 1\}$	:	correctness of the j-th word of **T**

To this end the following probability distribution functions (PDFs) are required and need to be estimated:

$$P(C_{\mathbf{S},\mathbf{T}} = 1|\mathbf{S}, \mathbf{T}) : \text{the probability that } \mathbf{T} \text{ is a correct translation of } \mathbf{S} \tag{1}$$

$$P(C_{\mathbf{S},\mathbf{T},j} = 1|\mathbf{S}, \mathbf{T}) : \text{the probability of correctness of the j-th word of } \mathbf{T} \text{ given that } \mathbf{T} \text{ is a translation of } \mathbf{S} \tag{2}$$

As **S** and **T** may be any sentence, directly estimating these probabilities is impossible. We therefore opted to map the pair $(\mathbf{S}, \mathbf{T})$ to a vector of d_s numerical features (so-called *predictive parameters* described in Sect. 7.1) via the function $\mathbf{x^s}$. Similarly $(\mathbf{S}, \mathbf{T}, j)$ were mapped to a numerical vector of d_w features via $\mathbf{x^w}$ (Sect. 6.1):

$$\mathbf{x^s} : (\mathbf{S}, \mathbf{T}) \in \mathcal{V}_S^* \times \mathcal{V}_T^* \rightarrow \mathbf{x^s}(\mathbf{S}, \mathbf{T}) \in \mathbb{R}^{d_s}$$

and:

$$\mathbf{x^w} : (\mathbf{S}, \mathbf{T}, j) \in \mathcal{V}_S^* \times \mathcal{V}_T^* \times \mathbb{N} \rightarrow \mathbf{x^w}(\mathbf{S}, \mathbf{T}, j) \in \mathbb{R}^{d_w}$$

Such parameters may include, for example, the length of source and target sentences, the score given by a translation model or a language model, etc. The following PDFs are thus learnt (the left-hand parts are just notations) instead of Formulae 1 and 2:

$$p(C_{\mathbf{S},\mathbf{T}}; \mathbf{S}, \mathbf{T}) \stackrel{def}{=} P(C_{\mathbf{S},\mathbf{T}} | \mathbf{x^s}(\mathbf{S}, \mathbf{T})) \tag{3}$$

$$p(C_{\mathbf{S},\mathbf{T},j}; \mathbf{S}, \mathbf{T}, j) \stackrel{def}{=} P(C_{\mathbf{S},\mathbf{T},j} | \mathbf{x^w}(\mathbf{S}, \mathbf{T}, j)) \tag{4}$$

Note that although it does not explicitly appear in the notation, p depends on the function $\mathbf{x}$, which will vary in different experiments, and will also not be the same on sentence- and word-levels. These distributions were to be learnt on large data sets (described in Sect. 4) by standard machine learning algorithms (Sect. 2) such as Support Vector Machines (Cortes and Vapnik 1995), Neural Networks (Fausett 1994), Logistic Regression (Menard 2002) or Partial Least Squares Regression (Tobias 1995).

1.3.1 Classification

After this training process the probability estimates (Formulae 3 and 4) could be used as confidence measures. It was then possible to compute a classification:

$$\hat{c} : (\mathbf{T}, \mathbf{S}) \rightarrow \hat{c}(\mathbf{T}, \mathbf{S}) \in \{0, 1\}$$

or at word-level::

$$\hat{c} : (\mathbf{T}, \mathbf{S}, j) \rightarrow \hat{c}(\mathbf{T}, \mathbf{S}, j) \in \{0, 1\}$$

In order to minimise the number of errors, classification needs to be performed according to:

$$\hat{c}(\mathbf{T}, \mathbf{S}) \stackrel{def}{=} \underset{c \in \{0,1\}}{\arg\max}\ p(c; \mathbf{S}, \mathbf{T}) \tag{5}$$

$$\hat{c}(\mathbf{T}, \mathbf{S}, j) \stackrel{def}{=} \underset{c \in \{0,1\}}{\arg\max}\ p(c; \mathbf{S}, \mathbf{T}, j) \tag{6}$$

However, this is too strict and neither accounts for biased probability estimates nor permits the attribution of levels of importance to correct rejection or correct acceptance, i.e. correct detection of good translations versus correct detection of erroneous translations (see performance estimation in Sect. 3). Therefore we introduced an *acceptance threshold* δ:

$$\hat{c}(\mathbf{T}, \mathbf{S}; \delta) \stackrel{def}{=} \begin{cases} 1 & \text{if } p(1; \mathbf{S}, \mathbf{T}) \geq \delta \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\hat{c}(\mathbf{T}, \mathbf{S}, j; \delta) \stackrel{def}{=} \begin{cases} 1 & \text{if } p(1; \mathbf{S}, \mathbf{T}, j) \geq \delta \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

If $\delta = 0.5$, then formulae 7 and 8 are equivalent to 5 and 6. However, setting a higher δ may compensate for a positive bias in probability estimates (3) and (4) or penalise false acceptances more heavily, while setting a lower δ compensates for a negative bias or penalises false rejections more heavily.

1.3.2 Bias

Probability estimates of Formulae 3 and 4 are often biased. This generally does not harm classification performance for two reasons. Firstly, when the bias is uniform ($p^* = \tilde{p} + b$ where b is constant), removing the bias is equivalent to setting an appropriate acceptance threshold. Secondly and most importantly, these PDFs are learnt by minimising classification cost. It is, therefore, unsurprising that even if the probabilities are biased, and even if the bias is not uniform ($p^* = \hat{p} + b(\hat{p})$), positive examples generally obtain a higher probability than negative ones.

However, biased probability estimates can harm other performance metrics and in particular will definitely harm Normalised Mutual Information (Sect. 3) as shown in Siu and Gish (1999). We thus estimated bias on a separate corpus as explained in the paper of Siu et al. The interval [0, 1] was split into 1000 non-overlapping bins $\mathcal{B}_i$ of uniform width, and bias was estimated separately on each of them.

$$\forall i \in \{1, \ldots, 1000\} \cdot b(\mathcal{B}_i) = \frac{\sum_{j|\hat{p}_j \in \mathcal{B}_i} \left(\hat{p}_j - c^*_j\right)}{\sum_{j|\hat{p}_j \in \mathcal{B}_i} 1} \tag{9}$$

where $\hat{p}_j$ are the estimated probabilities of the correctness of items in the training set dedicated to bias estimation, and c^*_j their true classes. Then we obtained an unbiasing function:

$$\text{if } p \in \mathcal{B}_i : unbias(p) = p - b(\mathcal{B}_i) \tag{10}$$

If $\hat{p}$ is the probability of correctness estimated by a confidence measure, we chose to use the unbiased estimation in our applications:

$$p(1; \mathbf{S}, \mathbf{T}) = unbias(\hat{p})$$

1.3.3 Sentence quality assessment

Some applications do not require the classification of sentences as correct or incorrect, but rather the estimation of overall quality of the translation. This would ressemble BLEU score (Papineni et al. 2002) or Translation Edit Rate (Snover et al. 2006) only without using reference translations. In this case a quality metric is more suitable than a correctness probability. In Sect. 7 we thus present a method for learning the PDF of Formula 3 which can also perform regression against quality scores. The training set for this task was:

$$\left\{\left(\mathbf{x}^{\mathbf{s}}(\mathbf{s}^n, \mathbf{t}^n); q^*_{\mathbf{s}^n,\mathbf{t}^n}\right)\right\}_{n=1\ldots N} \subset \mathbb{R}^{d_s} \times \mathbb{R}^+$$

where $q^*_{\mathbf{s}^n,\mathbf{t}^n}$ is a score relying on expert knowledge. This can be a human evaluation, or a metric computed by comparing the sentence to expert given references, like Word Error Rate, BLEU or Translation Edit Rate. The goal is to learn the mapping $f_\Theta : \mathbb{R}^{d_s} \rightarrow \mathbb{R}^+$ while minimising the mean quadratic error using regression techniques (e.g. linear regression, support vector regression, partial least squares regression) where Θ is a set of parameters to be estimated by regression:

$$\frac{1}{N}\sum_{n=1}^{N}\left|f_\Theta\left(\mathbf{x}^{\mathbf{s}}(\mathbf{s}^n, \mathbf{t}^n)\right) - q^*_{\mathbf{s}^n,\mathbf{t}^n}\right|^2 \tag{11}$$

1.3.4 Training sets

PDFs (Eqs. 3 and 4) and regression parameters Θ in Eq. 11 need to be learnt using large data sets. Such data sets consist of:

– N source sentences $\mathbf{s}^1, \ldots, \mathbf{s}^N$ which are realisations of $\mathbf{S}$.
– The corresponding N automatically translated sentences $\mathbf{t}^1, \ldots, \mathbf{t}^N$ which are realisations of $\mathbf{T}$.
– Reference sentences classes and a quality metric $\left(\left(c^*_{\mathbf{s}^1,\mathbf{t}^1}, q^*_{\mathbf{s}^1,\mathbf{t}^1}\right), \ldots, \left(c^*_{\mathbf{s}^N,\mathbf{t}^N}, q^*_{\mathbf{s}^N,\mathbf{t}^N}\right)\right)_{n=1,\ldots,N} \in \left(\{0,1\} \times \mathbb{R}^+\right)^N$ which are realisations of $C_{\mathbf{S},\mathbf{T}}$; they can be provided by human experts (Sect. 4.1) or automatically generated from human translations (Sect. 4.3 and 4.2).
– Reference word classes $\forall n \in \{1, \ldots, N\} \cdot \left(c^*_{\mathbf{s}^n,\mathbf{t}^n,1}, \ldots, c^*_{\mathbf{s}^n,\mathbf{t}^n,Len(\mathbf{t})}\right) \in \{0,1\}^{Len(\mathbf{t})}$ which are realisations of $C_{\mathbf{S},\mathbf{T},j}$ and also provided by human experts.

2 Classification and regression techniques

The problem of confidence estimation is now reduced to standard classification and regression problems. Many well-known machine learning approaches are available and we opted to experiment with often used techniques such as Support Vector Machines, Logistic Regression and Artificial Neural Networks, as well as with the less widely-known Partial Least Squares Regression.

2.1 Logistic regression

Here we wanted to predict the correctness $C \in \{0,1\}$ given a set of features $\mathbf{X} \in \mathbb{R}^d$; to this end we needed to estimate the distribution $P(C = 1|\mathbf{X})$. Logistic Regression (Menard 2002) consists of assuming that:

$$P(C = 1|\mathbf{X}) = \frac{1}{1 + e^{\langle \Theta, \mathbf{X} \rangle + b}} \tag{12}$$

for some $\Theta \in \mathbb{R}^d$ and $b \in \mathbb{R}$ and then optimise Θ with regard to the maximum likelihood criterion on the training data. Logistic regression was used not only to combine several features but also to map the scores produced by a confidence estimator to a probability distribution.

2.2 Support vector machines

The well-known Support Vector Machines (SVMs) (Hsu et al. 2003) have highly desirable characteristics which made them well-suited to our problem. They are able to discriminate between two non-linearly separable classes; they can also compute the probability that a given sample belongs to one class (and not only a binary decision), and they can also be used to perform regression against numerical scores (Smola and Schölkopf 2004). We used LibSVM (Chang and Lin 2011) for feature scaling, classification and regression.

2.2.1 SVM for classification

An SVM was trained to produce a probability of correctness. By doing so the acceptance threshold could be adapted (Sect. 1.3 and Eqs. 7 and 8), making the classifier more flexible. The kernel used was a Radial Basis Function since it is simple and was reported in Zhang and Rudnicky (2001) as giving good results:

$$K_\gamma(\mathbf{x}(\mathbf{s}, \mathbf{t}, j), \mathbf{x}(\mathbf{s}', \mathbf{t}', j')) = e^{-\gamma \|\mathbf{x}(\mathbf{s},\mathbf{t},j) - \mathbf{x}(\mathbf{s}',\mathbf{t}',j')\|^2}$$

2.2.2 SVM for quality evaluation

The same kernel was used but this time to perform regression against sentence-level BLEU score (Papineni et al. 2002).

2.2.3 Meta-parameters optimisation

SVMs require two meta-parameters to be optimised: the γ parameter of the radial basis function, and the error cost C. γ and C were optimised by grid search on the development corpus with regard to equal error rate for classification, and mean quadratic error for regression.

2.3 Neural networks

The FANN toolkit (Fast Artificial Neural Network (Nissen 2003)) is used for building feed-forward neural networks (NN). After experimenting on a development set we decided to stick to the standard structure, namely one input layer with as many neurons as we have features, one hidden layer with half as many neurons, and an output layer made of a single neuron which returns a probability of correctness. The connection rate was 0.5 in order to keep computation time tractable. We stuck to the default sigmoid activation function. The weights were optimised by standard gradient back-propagation.

2.4 Partial least squares regression

Partial Least Squares Regression (Wold et al. 1984; Specia et al. 2009) is a multivariate data analysis technique that finds a bilinear relation between the observable variables (our features $\mathbf{X}$ and the response variables, namely the probability of correctness $p(1; \mathbf{X})$ or the quality score). It works by projecting both predictors and observations on a linear subspace and performs least-squares regression in this space. It has the major advantage of being robust to correlated predictors.

3 Evaluation of the classifiers

Error rate is the most obvious metric for measuring the performance of a classifier. It is, however, not an appropriate metric because of its sensitivity to class priors (Kononenko and Bratko 1991; Siu and Gish 1999). Let us exemplify the problem and consider, for example, an MT system which gives roughly 15% of wrongly translated words. Now let us consider a confidence measure such that:

$$\forall \mathbf{s}, \mathbf{t}, j \; p^0(1; \mathbf{s}, \mathbf{t}, j) = 1$$

It makes no error on correct words (85% of total) but misclassifies all wrong words (15%). Its error rate is therefore $0 \times 0.85 + 1 \times 0.15 = 0.15$. Now let us consider a second confidence measure $p^1(1; \mathbf{s}, \mathbf{t}, j)$ which correctly detects every wrong word (if the j-th word of $\mathbf{t}$ is wrong then $p^1(1; \mathbf{s}, \mathbf{t}, j) = 0$) but also incorrectly assigns a null probability of correctness to 20% of the words that are appropriate translations. The error rate of this measure is: $0 \times 0.15 + 0.20 \times 0.85 = 0.17$.

p^0 thus seems to outperform p^1. This is, however, not true, because p^0 does not provide the user with any useful information (or strictly speaking, no information at all), while if $p^0(1; \mathbf{s}, \mathbf{t}, j) > 0$ then we would be certain that the word is correct. There is a lesson here. An appropriate metric for the usefulness of a confidence measure is not the number of misclassifications it makes but *the amount of information it provides*. This is why we opted to use *Normalised Mutual Information* (Siu and Gish 1999) to assess the performance of a measure (Sect. 3.2), along with Equal Error Rate (EER) and *Discrimination Error Trade-off* (DET) curves (Sect. 3.1). The latter is a powerful tool for the visualisation of the behaviour of a classifier with different acceptance thresholds and therefore different compromises between incorrect acceptances and incorrect rejections.

3.1 Discrimination error trade-off

A classifier makes two kinds of mistakes: *False acceptance* (or 'false positive', also called a *Type 1 error*), when an erroneous item (word or sentence) is classified as correct, and *False rejection* (or 'false negative', also known as a *Type 2 error*) when a correct item is classified as incorrect. When evaluating the performance of a classifier we know the predictions $\hat{c}$ (Eqs. 7 and 8) and the *actual* realisations c^* of the variables $\mathbf{C}$. As stated above in Sect. 1.3, $\hat{c}(\mathbf{t}; \mathbf{s}; \delta)$ is the *estimated* correctness of translation $\mathbf{t}$ given the source sentence $\mathbf{s}$ with acceptance threshold δ, and $c^*_{\mathbf{s},\mathbf{t}}$ is the *true* (expert-given) correctness (Sect. 1.3.4). The sentence-level false acceptance rate is:

$$e_1(\mathbf{s}, \mathbf{t}; \delta) = \begin{cases} 1 & \text{if } \hat{c}(\mathbf{t}; \mathbf{s}; \delta) = 1 \text{ and } c^*_{\mathbf{s},\mathbf{t}} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$$err_1(\delta) = \frac{\sum_{\mathbf{s},\mathbf{t}} e_1(\mathbf{s}, \mathbf{t}; \delta)}{\sum_{\mathbf{s},\mathbf{t}} \left(1 - c^*_{\mathbf{s},\mathbf{t}}\right)} \tag{14}$$

err_1 is thus the proportion of wrong items which are incorrectly accepted ($\sum_{\mathbf{s},\mathbf{t}} \left(1 - c^*_{\mathbf{s},\mathbf{t}}\right)$ is the number of wrong items).

The sentence-level false rejection rate is:

$$e_2(\mathbf{s}, \mathbf{t}; \delta) = \begin{cases} 1 & \text{if } \hat{c}(\mathbf{t}; \mathbf{s}; \delta) = 0 \text{ and } c^*_{\mathbf{s},\mathbf{t}} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

$$err_2(\delta) = \frac{\sum_{\mathbf{s},\mathbf{t}} e_2(\mathbf{s}, \mathbf{t}; \delta)}{\sum_{\mathbf{s},\mathbf{t}} c^*_{\mathbf{s},\mathbf{t}}} \tag{16}$$

err_2 is the proportion of correct items which are rejected by the classifier. Adapting these formulae to word-level is straightforward.

Intuitively err_1 is the proportion of erroneous words that the classifiers wrongly accept, while err_2 is the proportion of correct words that the classifier wrongly rejects. A relaxed classifier has a low err_2 and a high err_1, while a strict one has a low err_1 and a high err_2. Proof that err_1 and err_2 are insensitive to class priors was given in Siu and Gish (1999).

When δ goes from 0 to 1, more and more items are rejected. Accordingly, the false rejection rate (err_2) monotonically increases from 0 to 1, while the false acceptance rate (err_1) monotonically decreases from 1 to 0. The plot of $err_1(\delta)$ against $err_2(\delta)$ is called the *DET curve* (*Discrimination Error Trade-off*), cf. examples in Sect. 6.

A lower curve indicates a better classifier. All points of the DET curve should lie below the diagonal $[(0, 1), (1, 0)]$, which is the theoretical curve of a classifier using features uncorrelated with correctness (that is, inappropriate features).

Both err_1 and err_2 are generally approximations of continuous functions.[1] Thus a threshold δ_{EER} exists such that:

$$err_1(\delta_{ERR}) \simeq err_2(\delta_{EER}) = EER \tag{17}$$

EER is called the *equal error rate*. It can be seen as a 'summary' of the DET curve when the acceptance threshold is set so that there are the same proportions of Type 1 and 2 errors, and can be used for direct comparisons between classifiers. However, this is arbitrary, as the user may prefer to have fewer errors of one type, at the cost of more of the other type.

3.2 Normalised mutual information

Normalised Mutual Information (NMI) measures the level of informativeness of a predictive parameter or a set thereof in an application-independent manner (Siu and Gish 1999). Intuitively NMI measures the reduction of entropy of the distribution of true class C over the set $\{\text{“}correct\text{”}, \text{“}incorrect\text{”}\}$ when the value of the predictive parameter is known. Let $\mathbf{x}(\mathbf{S}, \mathbf{T})$ be a vector of predictive parameters:

$$\begin{aligned} NMI(C, \mathbf{x}) &= \frac{I(C; \mathbf{x})}{H(C)} = \frac{H(C) - H(C|\mathbf{x})}{H(C)} \\ H(C) &= -p^* log(p^*) - (1 - p^*) log(1 - p^*) \\ H(C|\mathbf{x}) &= \int_{\mathbf{v}} \Bigg(P(\mathbf{x}(\mathbf{S}, \mathbf{T}) = \mathbf{v}) \\ &\quad \times \sum_{c \in \{0,1\}} P(C = c|\mathbf{x}(\mathbf{S}, \mathbf{T}) = \mathbf{v}) log(P(C = c|\mathbf{x}(\mathbf{S}, \mathbf{T}) = \mathbf{v})) \Bigg) d\mathbf{v} \end{aligned} \tag{18}$$

where I is mutual information, H is entropy and p^* is the true prior probability of correctness. Since the true distribution $P(\mathbf{x}(\mathbf{S}, \mathbf{T}))$ is replaced with empirical frequencies observed in data, and $P(C|\mathbf{x}(\mathbf{S}, \mathbf{T}))$ is replaced with the computed estimation:

– Sentence-level NMI:

$$\begin{aligned} H(C|\mathbf{x}) \simeq \frac{1}{N} \sum_{(\mathbf{s},\mathbf{t}) \in \mathcal{S}} & (p(1; \mathbf{s}, \mathbf{t}) log(p(1; \mathbf{s}, \mathbf{t})) \\ & + (1 - p(1; \mathbf{s}, \mathbf{t})) log(1 - p(1; \mathbf{s}, \mathbf{t}))) \end{aligned} \tag{19}$$

[1] It actually depends on the true and estimated PDFs. When this is not the case, they will be approximated by continuous functions.

– Word-level NMI:

$$H(C|\mathbf{x}) \simeq \frac{1}{N_w} \sum_{(\mathbf{s},\mathbf{t})\in\mathcal{S}} \sum_{j=1}^{Len(\mathbf{t})} (p(1;\mathbf{s},\mathbf{t},j)log(p(1;\mathbf{s},\mathbf{t},j)) \\ +(1-p(1;\mathbf{s},\mathbf{t},j))log(1-p(1;\mathbf{s},\mathbf{t},j))) \quad (20)$$

$H(C|\mathbf{x})$ can never be lower than 0 and equality is achieved when for all pairs of sentences (or all words within such sentence-pairs), $p(c_{\mathbf{s},\mathbf{t}};\mathbf{s},\mathbf{t}) = 1$, which means that the true class is predicted with no uncertainty. On the other hand $H(C|\mathbf{x})$ can never be greater than $H(C)$, and equality is achieved when the predictive parameters are completely useless. Thus $M(\mathbf{x})$ is theoretically a real number between 0 and 1. However the approximation of $H(C|\mathbf{x})$ can be negative in practice.

4 Training and testing data

Large data sets are needed to learn PDFs of Formulae 3 and 4. Ideally a human professional translator would read the output of an MT system and assign a label (*correct* or *incorrect*) to each item. This method would give us high-quality training data but would be extremely expensive. Thus it would be preferable to use automatic or semi-automatic methods for efficiently classifying words and sentences. In the following we will discuss different methods for obtaining labelled data.

4.1 Expert-annotated corpora

This is the high-quality-high-cost whereby human experts analyse translations produced by an MT system and decide whether each word and sentence is correct or not. The classification depends on the application, but in our setting a word is classified as erroneous if it is an incorrect translation, if it suffers from a severe agreement error or if it is completely misplaced. A sentence is considered wrong if it is not clear that it has the same meaning as the sentence of which it is supposed to be a translation, or any meaning at all, or if it contains a significant level of ambiguity that was not apparent in the source sentence. This method has two major drawbacks. The first is that it is extremely slow and expensive, and the second is that it is not reproducible because a given sentence may be differently classified by different translators, or even by the same translator at different times.

We needed a small corpus of real, expert-annotated machine-translated sentences for our test set. To this end we set up the statistical MT system described as the baseline for WMT08 evaluation campaign following the instructions on the StatMT website[2]: it features a 5-gram language model with Kneser-Ney discounting trained with SRILM (Stolcke 2002) on about 35 million running words, IBM-5 translation model trained on around 40 million words, and Moses (Koehn et al. 2007) is used as the decoder. A held-out set of 40,000 sentence pairs was extracted from data for the purpose of

[2] http://statmt.org/wmt08/baseline.html.

training the confidence estimation system. We annotated a small set of 150 automatically translated sentences from transcriptions of news broadcast. Because of the spontaneous style of these sentences together with a vocabulary which did not match that of the training corpora (European Parliament), the BLEU score is not high (21.8 with only one reference). However, most translations were intelligible when given some thought.

A word was annotated as "incorrect" if it was completely irrelevant, very misplaced or grammatically flawed. Sentences were given scores ranging from one (hopelessly flawed) to five (perfect). For classification purposes we considered sentences scoring three or higher (possible to derive the correct meaning when given a little thought) to be correct.

Here are a few examples of expert-annotated sentences (the incorrect words are underlined):

Source sentence	Machine translation	score
je vous remercie monsieur le commis- -saire pour votre déclaration.	thank you mr commissioner for your <u>question</u>.	2
j'ai de nombreuses questions à poser à m. le commissaire.	i have <u>some</u> questions to ask to the commissioner.	4
les objectifs de la stratégie de lisbonne ne sont pas les bons.	the lisbon strategy mistaken.	3

4.2 Automatically annotated corpora

An intuitive idea is to compare a generated translation to a reference translation, and classify as correct the candidate words that are Levenshtein-aligned to a word in the reference translation (Ueffing and Ney 2004). However, this is too strict and many correct words would be incorrectly classified, because there are often many possible translations for a given source sentence and these may have nothing in common. This problem can partly be overcome by using multiple reference translations (Blatz et al. 2004). However multiple references are not always available and are costly to produce.

4.3 Artificial training data

In this section we present an algorithm which is aimed at obtaining the best of both worlds, namely automatically generating sentences (no humans involved, quickly generating huge amounts of data as with automatic annotation), and without any annotation error (no errors in gold standard classes as with human annotation). Our objective was to generate enough data for training classifiers in order to combine several predictive parameters.

Starting from human-produced reference translations, errors were automatically introduced in order to generate examples for training confidence measures. Given an English sentence **t** (a correct translation of source sentence **s**), we first chose where to introduce errors. As MT errors tend to be "bursty" (not evenly distributed but appearing in clusters), we implemented two error models whose parameters were estimated on a few human-annotated sentences. These annotations were not required to be extremely precise.

Bigram error model firstly we implemented a simple bigram model $P(C_i|C_{i-1})$, namely the probability that a word is correct given the correctness of the preceding word. The first word in a sentence has an a priori probability of being correct. According to this model we generated sequences of ones and zeroes corresponding to correct and incorrect words. We found that nine sentences out of ten in our human-annotated test set started with a correct word, that a wrong word had approximately a 50% chance of being followed by another wrong word ($P(C_i = 0|C_{i-1} = 0) \simeq 0.5$), and that a correct word had approximately a 90% chance of being followed by another correct word ($P(C_i = 1|C_{i-1} = 1) \simeq 0.9$).

Cluster error model the second explicitly models clusters. A sentence is a sequence of clusters of correct words and clusters of incorrect words: $\mathcal{C}_1, \ldots, \mathcal{C}_n$. By definition if a cluster contains correct words, the next cluster will contain incorrect words and vice versa. Let C_i be the correctness of words in the i-th cluster. $P(length(\mathcal{C}_\cdot)|C_\cdot = 0)$ and $P(length(\mathcal{C}_\cdot)|C_\cdot = 1)$ were estimated on a held-out set of 50 machine translations annotated by a human. Sequences of zeroes and ones were generated accordingly. The parameters of the model cannot theoretically be represented by a finite set of real numbers (they are distributions over $\mathbb{N}$). In practice, cluster lengths are bounded, so these distributions are actually over $\{0, \ldots, max(length((C_\cdot)))\}$. Just to give an idea, we found that the average length of a cluster of wrong words was 1.9 ($\sum_{k\geqslant 1} k \times P(length(\mathcal{C}_\cdot) = k|C_\cdot = 0) = 1.9$), with that of a cluster of correct words being 12.2.

Once the exact location of errors was known, we randomly introduced errors of five types: move, deletion, substitution, insertion and grammatical error. "Deletion" is straightforward: a word is chosen randomly according to a uniform distribution and deleted. "Move" is not much more complicated: a word is chosen at random according to the uniform distribution, and the distance it will be moved (jump length) is chosen according to a probability which is uniform within a given range (4 in our experiments) and null beyond. "Grammatical" errors are generated by modifying the ending of randomly selected words ("preserving" may become "preserved", "environment" may become "environmental"). "Substitution" and "insertion" are a little more subtle. Given the position i of the word to be replaced or inserted, the probability of every word in the vocabulary was computed using an IBM-1 translation model (Brown et al. 1993) and a 5-gram language model:

$$\forall t' \in \mathcal{V}_T \;\cdot\; p(t') = p_{IBM-1}\left(t'|\mathbf{s}_1^I\right) \times p_{5-gram}(t'|t_{i-4}, \ldots, t_{i-1})$$

The new word t' was then picked among all w at random according to the distribution p. This way the generated errors were not too "silly". WordNet (Miller 1995) was used to check that t' was not a synonym of t (otherwise it would not be an incorrect word): t' could not belong to any synset of which t is an element. The algorithm was controlled by several parameters, which were empirically chosen:

– probability distribution P_m of the proportion of move errors in a sentence and probability distribution P_j of jump length,
– probability distribution P_d of the proportion of deletions,
– probability distribution P_s of the proportion of substitutions,
– probability distribution P_i of the proportion of insertions,
– probability distribution P_g of the proportion of grammatical errors.

We chose triangle-shaped distributions with $mode = 0.2$, $minimum = 0$ and $maximum = 0.5$. These may not be the real distributions but seemed reasonable. The positions of words to be moved, deleted, inserted or modified were chosen according to uniform distribution probability. For each sentence errors were inserted in the order given previously; firstly, words were moved, then some were deleted, etc. Eventually we obtained a corpus with an average 16% word error rate, which approximately matches the error rate of real MT output.

Below is an example of degraded translation obtained using this method, extracted from our corpus:

Source sentence	Quant à eux, les instruments politiques doivent s'adapter à ces objectifs.
Reference translation	Policy instruments, for their part, need to adapt to these goals.
Degraded translation	Policy instruments, for the part, must to adapt to these goals.

We used 40,000 pairs of sentences (source: French-target: English) from the WMT-2008 evaluation campaign data. We degraded the reference translations according to the above rules. We found that the *bigram error model* gave the best results in the end (classification error rates of confidence measures trained on such data are lower) so we used it for all experiments presented here. The BLEU score of the degraded corpus was 56.5 which is much higher than the score of our baseline described in Sect. 4.1 (21.8). The latter score may be deemed to be an underestimation of the utility of our models since only one reference translation was available. However, this phenomenon did not affect the BLEU score of the degraded corpus as it came directly from the reference sentences, and so there was no need for multiple references. The error rate in the degraded corpus was set to 16% to match that of real MT output.

Others have proposed the use of artificial corpora, for example Blatz et al. (2004) and Quirk (2004). While we found that automatically generated corpora yield comparable performance to that of expert-annotated ones (Sect. 6.2), Quirk draws conclusions opposed to ours, as he found that a classifier trained on a small, human-annotated corpus performs better than one trained on a large automatically annotated corpora. However, in his experiments sentences are not automatically generated but automatically annotated. It is important to understand that automatically generated data is not the same as automatic annotation. In the latter, sentences are realistic but there is uncertainty redarding annotation. In contrast, while automatically degraded sentences may seem less realistic, there is almost no doubt that words labelled as incorrect are actually wrong, and vice versa. Thus automation plays a completely different role in the system of Quirk (2004) and ours. Another difference is that Quirk is evaluating sentences, while an important task for us is the evaluation of words. In Sect. 6.2 we present an experiment showing that a classifier trained on our large artificial corpus yields better results than one trained on a small human-annotated corpus (Fig. 4), for a fraction of the cost.

5 Experimental framework

A single feature (for example, *n*-gram probability) can be used as a confidence score. It is then relatively simple to evaluate its performance because no neural network or similar machine learning tool is necessary. Each word or sentence is attributed a score and a DET curve can be immediately computed. Computing NMI is a slightly more subtle operation because a probability is needed here, and not all predictive parameters qualify as such. In this case the score is turned into a probability by logistic regression (Sect. 2.1) whose parameters are learnt from artificial data.

Combining several predictive parameters is a little more complicated. Unless otherwise specified we proceeded as follows: two artificial corpora $\mathcal{T}_1$ (for "training") and $\mathcal{D}$ ("development") were used to find the best meta-parameters with regard to EER for SVM (γ and C, cf. Sect. 2.2) and Neural Networks (number of hidden units, cf. Sect. 2.3). Once optimal meta-parameters were found (or if none was set), the classifier was trained on a larger set of automatically generated data $\mathcal{T}_2$ and finally tested on real, unseen MT output $\mathcal{U}$. Then, if relevant, bias was estimated on a corpus of automatically generated data $\mathcal{B}$. $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{D}$ and $\mathcal{B}$ consisted of 10,000 sentences each (around 200,000 words). $\mathcal{U}$ consisted of 150 sentences, or approximately 3,000 words, with each of them having one reference translation (Sect. 4.1).

6 Word-level confidence estimation

We shall now look into the details of the predictive parameters we used (the components of the vector $\mathbf{x}(\mathbf{S}, \mathbf{T}, j)$) for word-level confidence estimation. These components will be noted x_{index} where $index$ is the label of the equation so that they are easier to find and refer to in the paper. Altogether these features are a numerical representation of a word in the target language (T_j), its context (the whole sentence $\mathbf{T}$), and the source sentence $\mathbf{S}$, the translation of which it is supposed to be a part. Of course this representation is less expressive than the original natural words and sentences, but hopefully it is more accessible to probability estimation while still bearing enough information to enable us to determine whether a word is correct or not.

Some of these features can themselves be used as confidence measures (for example LM-based features). In this case, we provided performance evaluation. Others cannot, such as Part-Of-Speech tag, stop word indicator and rule-based features.

6.1 Features for word-level confidence estimation

6.1.1 N-gram-based features

N-gram scores and backoff behaviour can provide a great deal of useful information. First, the probability of a word in a classical *3*-gram language model can be used as the feature:

$$x_{21}(\mathbf{S}, \mathbf{T}, j) = P\left(t_j | t_{j-1}, t_{j-2}\right) \tag{21}$$

Intuitively, we would expect an erroneous word to have a lower n-gram probability. However, this feature is generally already used in statistical MT systems, so the probability levels even of wrong words may not be too low.

Backward *3*-gram language models, proposed for speech recognition confidence estimation in Duchateau et al. (2002), also turned out to be useful:

$$x_{22}(\mathbf{S}, \mathbf{T}, j) = P\left(t_j | t_{j+1}, t_{j+2}\right) \tag{22}$$

This feature has the advantage of generally not being used in the decoding process.

Finally the backoff behaviour of the *3*-gram and backward *3*-gram models are powerful features: an n-gram not found in the language model may indicate a translation error. A score is given according to how many times the LM had to back off in order to assign a probability to the sequence, as proposed in Uhrik and Ward (1997) for speech recognition:

$$x_{23}(\mathbf{S}, \mathbf{T}, j) = \begin{cases} 1.0 & \text{if } t_{j-2}, t_{j-1}, t_j \text{ exists in the model} \\ 0.8 & \text{if } t_{j-2}, t_{j-1} \text{ and } t_{j-1}, t_j \text{ both exist in the} \\ & \text{model} \\ 0.6 & \text{if only } t_{j-1}, t_j \text{ exists in the model} \\ 0.4 & \text{if only } t_{j-2}, t_{j-1} \text{ and } t_j \text{ exist separately in} \\ & \text{the model} \\ 0.3 & \text{if } t_{j-1} \text{ and } t_j \text{ both exist in the model} \\ 0.2 & \text{if only } t_j \text{ exists in the model} \\ 0.1 & \text{if } t_j \text{ is completely unknown} \end{cases} \tag{23}$$

Figure 1 shows DET curves of the confidence measures based on *3*-grams and backward *3*-grams, together with scores and backoff behaviour. While *3*-grams and backward *3*-grams are almost indistinguishable, backoff behaviour performs better in terms of EER. Although this measure is very simple, it is less correlated with those used in the decoding or degrading process, which may explain why it achieves better discrimination results. The results are summarised in Table 1.

The NMI of backward *3*-gram scores is negative. This is theoretically not possible but may be explained by a strong bias in the estimation of probabilities which our unbiasing method was unable to efficiently remove (Sect. 1.3.2), and because NMI was only approximated here (Sect. 3.2).

6.1.2 Part-Of-Speech-based features

Replacing words with their POS class can help detect grammatical errors, and also take into account the fact that feature values do not have the same distributions for different word classes. Thus we used syntactic POS tags as a feature, along with the score of a word in a POS *3*-gram model. Tagging was performed using GPoSTTL, an open source alternative to TreeTagger (Schmid 1994, 1995).

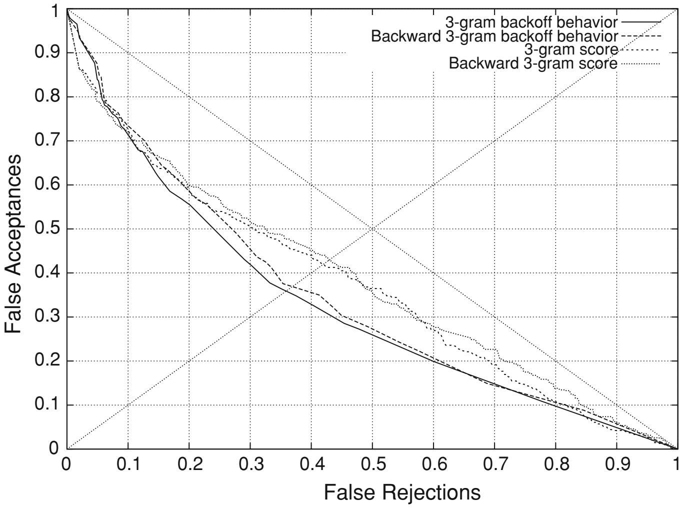

Fig. 1 DET curves of *3*-grams based confidence measures at word level

Table 1 Performance of *3*-gram-based confidence measures at word level

Feature	Equal error rate	Normalised mutual information
3-Grams	42.1	4.86×10^{-3}
Backward *3*-grams	42.9	-3.93×10^{-3}
Backoff	37.0	6.11×10^{-2}
Backward backoff	38.1	1.09×10^{-2}

$$x_{24}(\mathbf{S}, \mathbf{T}, j) = POS(t_j) \tag{24}$$

$$x_{25}(\mathbf{S}, \mathbf{T}, j) = P(POS(t_j)|POS(t_{j-2}), POS(t_{j-1})) \tag{25}$$

With our settings, POS is a non-numeric feature which can take 44 values, say $\{\pi_1, \ldots, \pi_{44}\}$. In order to combine it with numeric features, it was mapped to a vector $\pi(t_j) \in \{0, 1\}^N$ with N=40, as suggested in Hsu et al. (2003). The mapping is defined by

$$\pi(t_j)[i] = \begin{cases} 1 & \text{if } POS(t_j) = \pi_i \\ 0 & \text{otherwise} \end{cases}$$

We have chosen not to show the individual results of these confidence measures as they are only useful in combination with others.

6.1.3 Taking into account errors in the context

A common property of all *n*-gram-based features is that a word can receive a low score if it is actually correct but its neighbours are wrong. To compensate for this phenomenon, we took into account the average score of the neighbours of the word being con-

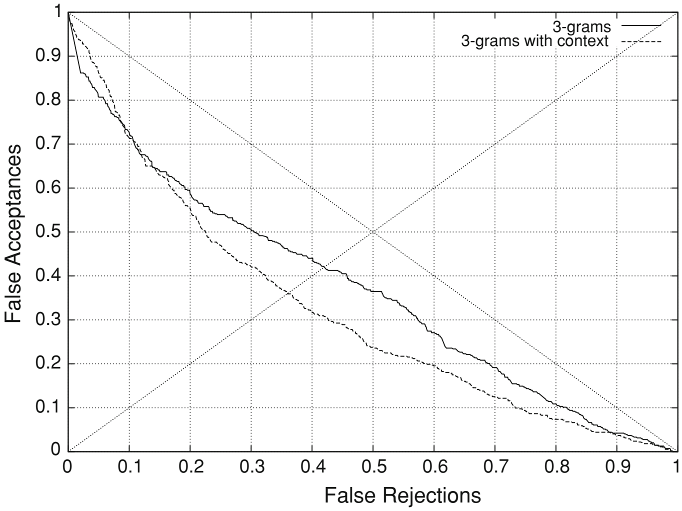

Fig. 2 DET curves of *3*-gram score combined with neighbours' score at word level

Table 2 Influence of taking the context into account

Bold values indicate that a paired T-test showed that this improvement is significant with a p-value of 1%

	Equal error rate	Normalised mutual information
3-Grams	42.1	4.86×10^{-3}
3-Grams and neighbours	36.3 (−**5.7**)	4.57×10^{-3}

sidered. More precisely, for every relevant feature $x_{.}$ defined above ($x_{21}, x_{22}, x_{23}, x_{25}$), we also computed:

$$x_{.}^{left}(\mathbf{S}, \mathbf{T}, j) = x_{.}(\mathbf{S}, \mathbf{T}, j-2) * x_{.}(\mathbf{S}, \mathbf{T}, j-1) * x_{.}(\mathbf{S}, \mathbf{T}, j)$$
$$x_{.}^{centred}(\mathbf{S}, \mathbf{T}, j) = x_{.}(\mathbf{S}, \mathbf{T}, j-1) * x_{.}(\mathbf{S}, \mathbf{T}, j) * x_{.}(\mathbf{S}, \mathbf{T}, j+1)$$
$$x_{.}^{right}(\mathbf{S}, \mathbf{T}, j) = x_{.}(\mathbf{S}, \mathbf{T}, j) * x_{.}(\mathbf{S}, \mathbf{T}, j+1) * x_{.}(\mathbf{S}, \mathbf{T}, j+2)$$

These predictive parameters were then combined using a neural network. Figure 2 and Table 2 show a vast improvement when using the product of *3*-gram probabilities of words in the centred window.

However, NMI was slightly harmed in the process. This may be because the product of *3*-gram scores on the window was not a proper estimation of probability of correctness. Nevertheless, it is perfectly possible to have a confidence measure with good discrimination power and a low NMI.

6.1.4 Intra-lingual mutual information

In Raybaud et al. (2009a,b) we introduced original predictive features based on mutual information, which is a metric for measuring how much information a random variable gives about another. Here we consider two random variables whose realisations are words, say W_1 and W_2:

$$I(W_1, W_2) = \sum_{w_1, w_2} P(W_1 = w_1, W_2 = w_2) \times log\left(\frac{P(W_1 = w_1, W_2 = w_2)}{P(W_1 = w_1)P(W_2 = w_2)}\right)$$

We used point-wise mutual information which is the contribution of a specific pair of words to the mutual information between W_1 and W_2 (that is, a single term of the sum above).

$$MI(w_1, w_2) = P(W_1 = w_1, W_2 = w_2) log\left(\frac{P(W_1 = w_1, W_2 = w_2)}{P(W_1 = w_1)P(W_2 = w_2)}\right)$$

The tuple $(w_1, w_2, MI(w_1, w_2))$ is called a *trigger*. Triggers are learnt on an unaltered bilingual corpus. The idea of using mutual information for confidence estimation was first expressed in Guo et al. (2004). It has since been proved useful for computing translation tables (Lavecchia et al. 2007).

Intra-lingual mutual information (IMI) measures the level of similarity between the words in a generated sentence, thus assessing the consistency of the sentence. Formally W_1 and W_2 are any T_i and T_j here (words in the translation hypothesis). Let J be the length of the translation hypothesis. The feature for confidence estimation is:

$$x_{26}(\mathbf{S}, \mathbf{T}, j) = \frac{1}{J-1} \sum_{1 \le i \ne j \le J} MI(t_i, t_j) \tag{26}$$

6.1.5 Cross-lingual mutual information

Cross-lingual mutual information (CMI) is similar to the previous intra-lingual mutual information in that it assesses source-translation consistency. Let I be the length of the source sentence:

$$x_{27}(\mathbf{S}, \mathbf{T}, j) = \frac{1}{I} \sum_{1 \le i \le I} MI(s_i, t_j) \tag{27}$$

Here W_1 and W_2 are any S_i and T_j.

Table 3 summarises the performance of MI-based features when used as confidence measures by themselves. Although they perform poorly, we will see that they are useful when combined with other predictive parameters (Sect. 6.2).

Table 3 Performance of mutual information-based features at word level

Feature	Equal error rate	Normalised mutual information
Intra-lingual	45.8	9.46×10^{-4}
Cross-lingual	45.7	-2.21×10^{-1}

Table 4 Performance of IBM-1-based confidence measure at word level

Feature	Equal error rate	Normalised mutual information
IBM-1 score	45.0	-1.84×10^{-3}

6.1.6 IBM-1 translation model

This feature was proposed in Blatz et al. (2004), Ueffing and Ney (2005):

$$x_{28}(\mathbf{S}, \mathbf{T}, j) = \frac{1}{I+1} \sum_{i=0}^{I} p_{IBM-1}(t_j|s_i) \tag{28}$$

where s_0 is the empty word. The performance of this predictive parameter used alone is given in Table 4. Once again the results are disappointing. The results are extremely similar to alignment probability (the sum is replaced by a max). It is surprising to note that even on a translation evaluation task, measures involving only the hypothesis yield better performance than those taking the source sentence into account.

Like MI-based features, IBM-1 does not work very well when used as a confidence measure and will only be used in combination with others.

6.1.7 Stop words and rule-based features

The "stop word" predictive parameter is a simple flag indicating whether the word is a stop word (*the, it*, etc.) or not. It helps a classifier to take into account the fact that the distribution of other features is not the same for stop words compared to content words. This feature is less informative than Part-Of-Speech, but simpler.

$$x_{29}(\mathbf{S}, \mathbf{T}, j) = \begin{cases} 1 & \text{if } t_j \text{ is a stop word} \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

The stop list was generated by picking words that are both short and frequent. Finally, we implemented four binary features indicating whether the word is a punctuation symbol, numerical, a URL or a proper name (based on lists of each type). These features were of course not designed to be used as standalone confidence measures.

6.2 Feature combination

Altogether we had 66 features for word-level confidence estimations, many of them very similar (for example *3*-gram probability and average *3*-gram probabilities on different windows), some very crude (for example sentence-level features like length ratio (cf. Sect. 7.1.5) used at word level). We trained four classifiers (Logistic Regression, Partial Least Squares Regression, Support Vector Machines and Neural Networks) to discriminate between correct and incorrect words based on these features. Only Neural Networks gave a consistent improvement over the best feature used alone

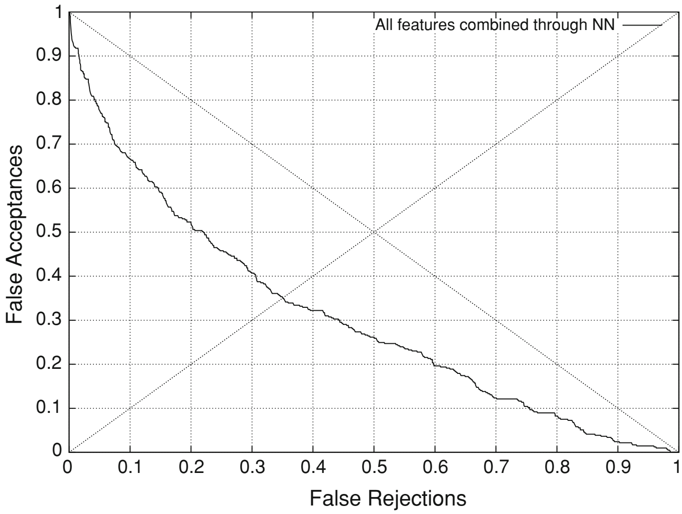

Fig. 3 Combination of all features by neural network

Table 5 Performance of all word-level features combined

Classifier	Equal error rate	NMI	Training time	Testing time
Logistic regression	36.8	-2.61×10^{-2}	13″	5″
PLSR	37.5	-5.84×10^{-2}	15′	1″
SVM	36.7	-1.87×10^{-1}	12h	500″
Neural network	35.0	6.06×10^{-2}	10′	2″

(*3*-gram scores on a centred window, cf. Sects. 6.1.1 and 6.1.3) for the classification task, although this was not a large improvement (−1.3 EER points). The DET curve for neural networks is presented in Fig. 3 and the results are summarised in Table 5.

The network used was a fully connected three-layer perceptron with 66 input nodes, 33 hidden nodes and one output node. The activation function is sigmoid.

The NMI results were especially disappointing. As explained in Sect. 3.2, NMI is harmed by bias. Although we estimated bias on a dedicated set of training data and removed it from the final estimation, we believe that the poor performance may perhaps be explained by the fact that bias is very different for artificial and natural data and probably much more important on the latter.

In order to evaluate the performance gain given by the automatically generated training corpus, we also split the annotated sentences into a training set (70 sentences), a development set (30 sentences) and a test set (50 sentences), on which we trained and evaluated the neural network. Figure 4 and Table 6 show that training on annotated data does not yield better results than training on the generated corpus. The natural corpus is small, but it must be noted that the artificial corpus was generated in just a few hours, while it took more than one day to annotate all the sentences. In addition, human annotations are subject to time and inter-annotator variations. Employing a

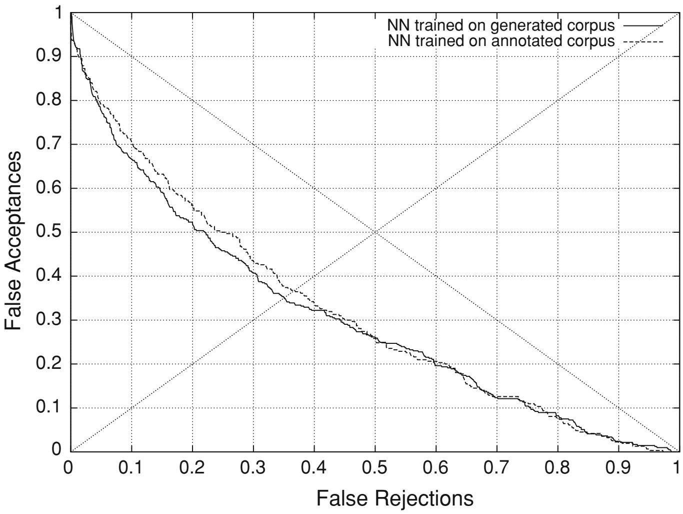

Fig. 4 Training neural network on annotated or generated corpus

Table 6 Performance of all word-level features combined

Classifier	Equal error rate	NMI
NN trained on generated corpus	35.0	6.06×10^{-2}
NN trained on annotated corpus	36.8	5.79×10^{-2}

Table 7 Contribution of mutual information-based confidence measure to overall performance

	Equal error rate	NMI
Without IMI and CMI	35.6	5.32×10^{-2}
With IMI and CMI	35.0	6.06×10^{-2}
Improvement	$\mathbf{-0.60}$	$\mathbf{+7.4 \times 10^{-3}}$

Bold values indicate that a paired T-test showed that this improvement is significant with a p-value of 1%

trained professional may alleviate these problems but this, of course, would be more expensive.

In Table 7 we show the modest contribution of mutual information (Sects. 6.1.4 and 6.1.5) to the performance of neural network combination of the features.

7 Sentence-level confidence estimation

The features described in this Section form a numerical representation of a pair made up of a source sentence and a target sentence. As in the previous section, our aim was to compute the distribution of probability of correctness on the numerical space (a subspace of $\mathbb{R}^{d_{sentence}}$). Unlike at the word level, the algorithm for generating degraded sentences cannot reliably tell if a degraded sentence is still correct or not. We circumvented the problem of creating a corpus for training classifiers (Sect. 7.2) but we could not automatically generate a corpus for estimating probability biases. Thus all normalised mutual information is poor.

Many word-level features can be extended to the sentence level by arithmetic or geometric averaging, e.g. IBM-1 translation probability, n-gram probability, etc.

7.1 Features for sentence-level confidence estimation

7.1.1 LM-based features

The first features we propose are sentence normalised likelihood in a *3*-gram model (forward and backward) and average backoff behaviour:

$$x_{30}(\mathbf{s}, \mathbf{t}) = \left(\prod_{j=1}^{J} P(t_j | t_{j-1}, \ldots, t_{j-n+1})\right)^{\frac{1}{J}} \tag{30}$$

$$x_{31}(\mathbf{s}, \mathbf{t}) = \left(\prod_{j=1}^{J} P(t_j | t_{j+1}, \ldots, t_{j+n-1})\right)^{\frac{1}{J}} \tag{31}$$

$$x_{32}(\mathbf{s}, \mathbf{t}) = \frac{1}{J} \sum_{j=1}^{J} x_{23}(\mathbf{S}, \mathbf{T}, j) \tag{32}$$

These features can also be used as confidence measures by themselves and their performance as such is presented in Table 8 and Fig. 5 together with intra-lingual mutual information, another kind of language model.

The following predictive parameter is the source-sentence likelihood. Its aim is to reflect how difficult the source sentence is to translate. It is obviously not designed to be used alone.

$$x_{33}(\mathbf{s}, \mathbf{t}) = \left(\prod_{i=1}^{I} P(s_i | s_{i-1}, \ldots, s_{i-n+1})\right)^{\frac{1}{I}} \tag{33}$$

Table 8 Performance of *3*-gram- and backoff-based confidence measures at sentence level

Feature	Equal error rate	Normalised mutual information
3-Gram normalised likelihood	41.7	4.02×10^{-3}
Backward *3*-gram normalised likelihood	41.3	3.97×10^{-3}
Averaged backoff behaviour	34.2	4.15×10^{-3}

7.1.2 Average mutual information

$$x_{34}(\mathbf{s},\mathbf{t}) = \frac{1}{J \times (J-1)} \sum_{i=1}^{J} \sum_{1 \le j \ne i \le J} MI(t_i, t_j)$$
$$= \frac{1}{J} \sum_{j=1}^{J} x_{26}(\mathbf{s},\mathbf{t}, j) \quad (34)$$
$$x_{35}(\mathbf{s},\mathbf{t}) = \frac{1}{I \times J} \sum_{i=1}^{I} \sum_{j=1}^{J} MI(s_i, t_j)$$
$$= \frac{1}{J} \sum_{j=1}^{J} x_{27}(\mathbf{s},\mathbf{t}, j) \quad (35)$$

We were surprised to observe that cross-lingual MI performed even worse at sentence level than at the word level. We have only presented the results for intra-lingual MI in Fig. 5 and Table 9, as its performance was closer to other standard confidence measures than it was at word level.

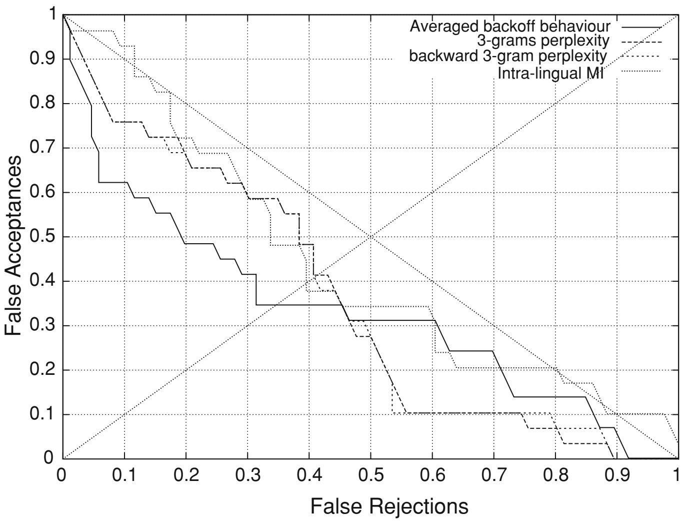

Fig. 5 DET curves of *3*-gram-, backoff- and intra-lingual mutual information-based confidence measures at sentence level

Table 9 Intra-lingual mutual information CM as a sentence-level confidence measure

Feature	Equal error rate	Normalised mutual information
IMI	39.0	9.46×10^{-4}

7.1.3 Normalised IBM-1 translation probability

The score of a sentence is its translation probability in IBM model 1, normalised to avoid penalising longer sentences:

$$x_{36}(\mathbf{s}, \mathbf{t}) = \left(\prod_{i=1}^{I} \sum_{j=0}^{J} P(s_i | t_j) \right)^{\frac{1}{I}} \tag{36}$$

As was the case at word level, it is surprising to note that although the system was tested on a translation task, confidence measures involving the source sentence do not perform better than those involving only the target sentence.

7.1.4 Basic syntax check

A very basic parser checks that brackets and quotation marks are matched, and that full stops, question or exclamation marks, colon or semi-colon are located at the end of the sentence (Blatz et al. 2004).

$$x_{37}(\mathbf{s}, \mathbf{t}) = \begin{cases} 1 & \text{if } \mathbf{t} \text{ is parsable} \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

This feature and the following are only pieces of information about the source and target sentences; they are not confidence measures themselves.

7.1.5 Length-based features

These very basic features reflect levels of consistency between the lengths of a source sentence and its translation (Blatz et al. 2004). The idea is that source and target sentences should be approximately of the same length, at least for language pairs such as French/English:

$$x_{38}(\mathbf{s}, \mathbf{t}) = Len(\mathbf{s}) \tag{38}$$

$$x_{39}(\mathbf{s}, \mathbf{t}) = Len(\mathbf{t}) \tag{39}$$

$$x_{40}(\mathbf{s}, \mathbf{t}) = \frac{Len(\mathbf{t})}{Len(\mathbf{s})} \tag{40}$$

7.2 Combination of sentence-level features

As explained earlier in the paper, a generation algorithm cannot tell which sentences are to be considered correct and which are not. Therefore, for sentence-level confidence, it was not directly possible to train classifiers to discriminate between correct and incorrect sentences. Instead, we used SVM, Neural Networks and Partial Least

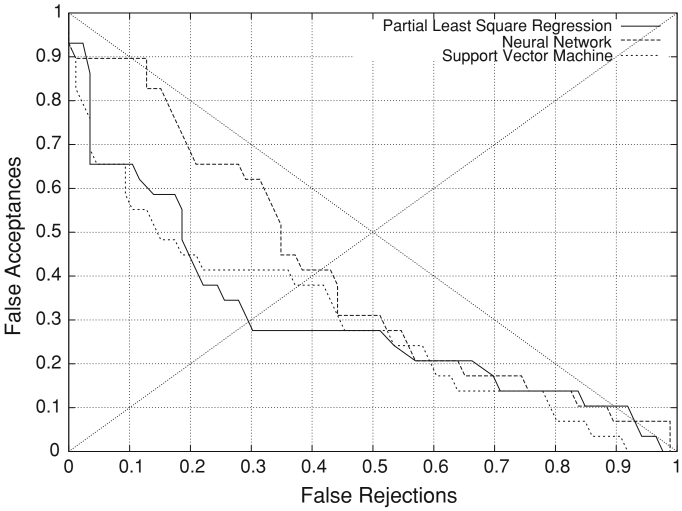

Fig. 6 DET curves of PLS and Neural Network combination of sentence-level features

Table 10 Performance of PLS, SVM and neural nets at sentence-level

Feature	Equal error rate	NMI
PLS	29.0	8.14×10^{-2}
SVM	38.0	-2.56×10^{-1}
Neural net	41.3	-2.44×10^{-2}

Squares (PLS) to perform regression against sentence-level BLEU score.[3] Sentences were then classified by thresholding this score (Fig. 6; Table 10).

Only PLS was found to improve (by 5.2 points, absolute) on the best standalone confidence measure (Average backoff behaviour, Sect. 7.1.1). Its correlation coefficient with human evaluation was 0.358.

8 Preliminary post-editing experiment

The previous sections have given a detailed explanation of how the proposed confidence measures work and the amount of errors they are able to detect. In this section we will describe a more subjective usability experiment. Our aim was to obtain qualitative feedback from real users of the system about the usability of confidence measures for assisted post-editing. Because of the limited number of subjects, and the fact that many predictive parameters are still work-in-progress, these results are only to be interpretated as hints regarding what users want and find useful, what we did right or

[3] It is true that BLEU is not very suited for sentence-level estimation. It has the advantage of being a well known automatic metric for which efficient toolkits are available. We also experimented with TER (Snover et al. 2006) but too many sentences produced a null score.

wrong and which direction we should follow in our research. The experimental protocol is inspired by the one described in Plitt and Masselot (2010). We implemented a post-editing tool with confidence measures and let users correct machine-translated sentences, with and without the help of confidence measures.

8.1 The post-editing tool

The program we developed (see screenshot in Fig. 7) can be seen as a simplified version of a tool for Computer-Assisted Translation. It displays a source sentence (in our case, in French) and a translation (in English) generated by Moses (Koehn et al. 2007). Errors detected by the confidence measures are highlighted. The user can then opt to edit the proposed translation.

The source sentence is displayed in the top field with the candidate translation in the field below. On the left there is a slider with which the user can change the acceptance threshold of the confidence estimation system (cf. Sect. 1.3.1). All words with a score below this threshold are displayed in red. Simplified explanations are given to the user, who does not require a full 'lecture' on confidence estimation: s/he is told that s/he may use an automatic help to detect erroneous words, and that the requested quality can be changed with this slider, if s/he so wishes. Of course, if his/her quality requirements are too high (corresponding to a threshold value of 1, i.e. the point to the far right on the DET curve, cf. Sect. 3.1), the system will incorrectly consider all words to be wrong. The user can edit the candidate translation if s/he thinks it is necessary. When s/he is satisfied with the translation, s/he has to click on "next". For the sake of the experiment the user may not come back to a sentence that has already been validated. If required, the user can click on "pause" to take a break, thus avoiding the problem of the program continuing to count the time spent on the translation, which would cause the time statistics to be meaningless. However, none of the users ever took a break. Everything else on this GUI is cosmetic (progress bar, etc.).

The total time spent on each sentence was recorded (the time between the loading of the sentence and clicking on the "next" button). This is actually the sum of three partial times, which are also recorded: time typing on the keyboard, time spent on the interface (moving the acceptance slider) and thinking time (the rest).

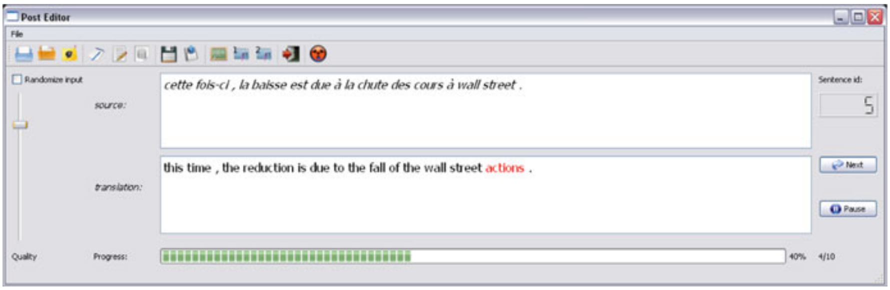

Fig. 7 Screenshot of the post-editing software

It should be noted that the proposed translation and confidence scores were not computed on-the-fly, in order to keep the program responsive and easily portable. This is quite a heavy constraint because the system cannot take the user's edits into account to compute a new, improved translation, and cannot compute the confidence of the post-edited translation (our users were of course informed of that). Furthermore, while all users stated that the program was easy to use, an ergonomist's input would be required to ensure that we made the right choices with regard to usability and that what we measure is really the influence of confidence measures and is not due to influence of the interface.

8.2 Experimental protocol

Since we were not expecting many volunteers, we wanted their English skills to be as homogeneous as possible (all of them are French native speakers) in order to limit the variability of the results. Seven subjects volunteered for the experiment. Six of them are English teachers and one is a master student in English. Unfortunately two of them failed to correctly follow the instructions and the corresponding data was discarded. The experiment lasted approximately two hours, divided into four stages:

First stage: introduction and training The users were provided with some basic explanations about the domain and the task and given ten sentences to post-edit along with simple instructions (see below). These sentences were just for training purposes and were not included in the final results.

Second stage: first experiment The users were told to start the first experiment when ready. They were given 30 sentences with their corresponding MT output and were told they could post-edit these translations with the help of the confidence measures.

Third stage: second experiment This experiment was identical to the first, except that the users did not have access to confidence measures. One volunteer out of two had the second experiment before the first, in order to compensate for the "training effect" (users complete the second experiment faster than the first one) and for fatigue (a user may be tired by the time s/he starts the second experiment, thus affecting post-editing speed and quality).

Fourth stage: user feedback Finally, the users were asked to complete a questionnaire, providing us with feedback on the post-editing software and the confidence measures.

We gave the following instructions to the users, with the idea that translated documents must be good enough to be read without extra effort, but not necessarily in perfectly idiomatic English:

- The goal is to obtain a correct translation, not necessarily a very fluent one. Fix mistakes, not style.
- You can use any help you want (most of them actually used paper or online dictionaries) but:
  - Don't use an online tool to re-translate the sentence

– Don't spend too much time on details
– Don't ask the supervisor for help

The sentences were random subsets of the test set of the WMT09 campaign, which comprises transcripts of news broadcast. Each user had to post-edit two randomized sets of thirty sentences. This choice is questionable insofar as most 'real life' applications consist of translating whole documents and not a sequence of sentences without connections to each other. However, we chose randomized subsets so that the intrinsic difficulty of the task did not influence the results.

8.3 Results and analysis

Table 11 summarises the most important results of the experiments. Most of these metrics are straightforward but some are worthy of more explanation.

Sentence quality After the experiment, all the post-edited translations were scored by a team member, a native French speaker also fluent in English. Each sentence received a score between 1 and 5 in the same fashion as in StatMT evaluation tasks:

1. the translation is completely unusable.
2. the translation is seriously faulty but a degree of meaning can be grasped.
3. the translation is usable although not very good.
4. the translation has minor flaws.
5. the translation is very good.

Correlation between confidence estimations and edits our aim here was to check how the user's decisions and the machine predictions correlated. To this end every word in the machine-generated hypothesis was mapped to 1 if it was Levenshtein-aligned to a word in the edited hypothesis (which means it was not modified), and 0 otherwise (which means it had been inserted or modified by the user). The corpus was, there-

Table 11 Effect of confidence estimation on a post-editing task

	Without CM	With CM
Average time per sentence (s)	77	87
Average edit rate	30%	32%
Average sentence quality	4.3	4.2
	First experiment	Second experiment
Average time per sentence	84.22	80.12
Average edit rate	0.29	0.33
Average sentence quality	4.2	4.3
Ratio of corrections/detected errors	1.76	
Correlation between CM and edits	0.23	

CM confidence measure

fore, mapped to a sequence of 0 and 1 and we computed the correlation between this sequence and the estimated probabilities of correctness.

Ratio of number of edits over number of detected errors this is the ratio of the number of edits made to the original hypothesis over the number of errors which were detected by the system. A high ratio suggests that the user could not find an appropriate trade-off between false positives and false negatives and had to lower his/her quality requirement (using the slider) in order to obtain an acceptable level of accuracy.

While the results in terms of translation speed are disappointing (Table 11), this experiment was primarily designed to obtain a qualitative feedback from real users of the system. This is what the following analysis will focus on, in order to determine what must be improved and how. A more fine-grained analysis showed that the time difference is entirely due to “thinking” time. User feedback confirmed that they thought the help was not reliable enough to be useful, and that even if it sometimes drew their attention to some mistakes, checking the systems’ recommendations wasted too much of their time. However, it must be noted that users were significantly faster during the second post-editing task than the first. This suggests that more training is needed before users would grow accustomed to the task and really see the program as a tool instead of a constraint. We believe that an experiment involving more users over a longer time frame is necessary. The consistently high and comparable edit rate with and without confidence measures suggests—and this is confirmed by feedback—that a lot of editing was required, but the high ratio of number of corrections over automatically detected errors suggests that confidence measures were not able to precisely discriminate between correct and incorrect words. Regardless of confidence estimation, many of our users stated that they would rather translate a sentence from scratch than edit flawed MT output.

As a conclusion to this experiment, we propose the following directions for further improvements and experiments:

- The users should be given a consistent task, not random sentences.
- Users need a longer amount of training time as some of them were still not sure what to do with the slider by the end of the tasks. Measurements show that their efficiency continued to increase after the training stage. We believe they need more time to familiarise themselves with the tool and make the best use of it.
- The program interface needs to be carefully designed with ergonomics in mind in order to really measure the influence of confidence measures and not that of the GUI.
- We need more reliable confidence measures and above all, we greatly need to focus on precision rather than recall as we observed that false alarms were very disconcerting for users.

9 Conclusion

After introducing and formalising the problem, we presented a method which makes it possible to generate large amounts of training data. We then developed a list of predictive parameters which we consider are some of the most significant for confidence estimation, including two original measures based on mutual information. We compared different machine learning techniques combining the features we proposed. From these features, we consider Neural Networks and Partial Least Squares Regression to be the best suited, depending on the application. We have shown that combining many features improves over the best predictive parameters alone, by 1.3 points (absolute) EER at word level and 6 points at sentence level on a classification task. Finally, we presented an experiment aimed at measuring how helpful confidence estimation is in a post-editing task. This experiment suggested that our confidence estimation system is not mature enough to be helpful in such a setting. However, the limited number of volunteers and the lack of long-term observations makes the results somewhat difficult to interpret. Nevertheless, the knowledge we gained from this experiment and users feedback will help us improve confidence measures for the benefit of future users.

Our hope is that this paper will provide the necessary information to enable the construction of a complete confidence estimation system for MT from scratch and facilitate the incorporation therein of new predictive features. In addition to assisted post-editing, we believe there are many useful applications for confidence estimation, namely:

- Warning a user that the translation s/he requested may be flawed,
- Automatically rejecting hypotheses generated by the decoder or combining several systems in a voting system,
- Recombining good phrases from an n-best list or a word graph to generate a new hypothesis.

We have also identified important research directions in which this work could be extended to make confidence measures more helpful for users. Firstly, we would cite computing confidence estimates at phrase level which would enable users to work on semantically consistent chunks while retaining a more fine-grained analysis than with sentences. Secondly, semantic features could be introduced which would make it possible to detect otherwise tricky errors such as missing negations, and help users to focus on errors of meaning rather than grammatical errors and disfluencies which are, in some cases, arguably less important.

Acknowledgements We would like to warmly thank all students and staff of IUFM of Lorraine (http://www.lorraine.iufm.fr) who volunteered for the post-editing experiment and made the experiment possible. We would particularly like to thank Mr. Gilles Grateau for his careful and conscientious help throughout this work and for his patience and constant cheerfulness.

References

Blatz J, Fitzgerald E, Foster G, Gandrabur S, Goutte C, Kulesza A, Sanchis A, Ueffing N (2004) Confidence estimation for machine translation. In: 20th International conference on computational linguistics, proceedings, Vol I. Geneva, Switzerland, pp 315–321

Brown P, Della-Pietra S, Della-Pietra V, Mercer R (1993) The mathematic of statistical machine translation: parameter estimation. Comput Linguist 19(2):263–311

Chang C-C, Lin C-J (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297

Duchateau J, Demuynck K, Wambacq P (2002) Confidence scoring based on backward language models. In: Proceedings of IEEE international conference on acoustics, speech, and signal processing, Vol. 1. Orlando, Florida, pp 221–224

Fausett LV (1994) Fundamentals of neural networks. Prentice-Hall, Englewood Cliffs

Gandrabur S, Foster G (2003) Confidence estimation for translation prediction. In: HLT-NAACL 2003: conference combining human language technology conference series and the North American chapter of the association for computational linguistics conference series. Vol. 4. Edmonton, Canada, pp 95–102

Guo G, Huang C, Jiang H, Wang R (2004) A comparative study on various confidence measures in large vocabulary speech recognition. In: Proceedings of the international symposium on chinese spoken language processing. Hong Kong, China, pp 9–12

Hsu C-W, Chang C-C, Lin C-J (2003) A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University, Taiwan

Koehn P, Hoang H, Birch A, Callison-Burch C, Federico M, Bertoldi N, Cowan B, Shen W, Moran C, Zens R et al (2007) Moses: Open source toolkit for statistical machine translation. In: ACL 2007, Proceedings of the interactive poster and demonstration sessions. Prague, Czech Republic, pp 177–180

Kononenko I, Bratko I (1991) Information-based evaluation criterion for classifier's performance. Mach Learn 6(1):67–80

Lavecchia C, Smaïli K, Langlois D, Haton J-P (2007) Using inter-lingual triggers for Machine translation. In: Proceedings of the eighth international conference on speech communication and technology (INTERSPEECH). Antwerp, Belgium, pp 2829–2832

Menard SW (2002) Applied logistic regression analysis, Sage university papers, Quantitative applications in the social sciences. Sage, Thousand Oaks, CA

Miller G (1995) WordNet: a lexical database for English. Commun ACM 38(11):39–41

Nissen S (2003) Implementation of a fast artificial neural network library (fann). Technical report, Department of Computer Science University of Copenhagen (DIKU), Copenhagen, Denmark. http://fann.sf.net

Papineni K, Roukos S, Ward T, Zhu W (2002) Bleu: a method for automatic evaluation of machine translation. In: 40th Annual meeting of the association for computational linguistics. Philadelphia, USA, pp 311–318

Plitt M, Masselot F (2010) A productivity test of statistical machine translation post-editing in a typical localisation context. Prague Bull Math Linguist 93(1):7–16

Quirk C (2004) Training a sentence-level machine translation confidence measure. In: LREC-2004: Fourth international conference on language resources and evaluation, proceedings. Lisbon, Portugal, pp 825–828

Raybaud S, Lavecchia C, Langlois D, Smaïli K (2009a) New confidence measures for statistical machine translation. In: Proceedings of the international conference on agents and artificial intelligence. Porto, Portugal, pp 61–68

Raybaud S, Lavecchia C, Langlois D, Smaïli K (2009b) Word- and sentence-level confidence measures for machine translation. In: EAMT-2009: Proceedings of the 13th annual conference of the european association for machine translation. Barcelona, Spain, pp 104–111

Schmid H (1994) Probabilistic part-of-speech tagging using decision trees. In: Proceedings of international conference on new methods in language processing, Vol. 12. Manchester, UK, pp 44–49

Schmid H (1995) Improvements in part-of-speech tagging with an application to German. In: In Proceedings of the EACL SIGDAT-workshop. Dublin, Ireland, pp 47–50

Simard M, Ueffing N, Isabelle P, Kuhn R (2007) Rule-based translation with statistical phrase-based post-editing. In: Proceedings of the ACL-2007 workshop on statistical machine translation (WMT-07). Prague, Czech Republic, pp 203–206

Siu M, Gish H (1999) Evaluation of word confidence for speech recognition systems. Comput Speech Lang 13(4):299–318

Smola A, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14(3):199–222

Snover M, Dorr B, Schwartz R, Micciulla L, Makhoul J (2006) A study of translation edit rate with targeted human annotation. In: AMTA 2006: Proceedings of the 7th conference of the association for machine translation in the Americas: visions for the future of machine translation. Cambridge, MA, USA, pp 223–231

Specia L, Cancedda N, Dymetman M, Turchi M, Cristianini N (2009) Estimating the sentence-level quality of machine translation systems. In: EAMT-2009: Proceedings of the 13th annual conference of the European association for machine translation. Barcelona, Spain, pp 28–35
Stolcke A (2002) SRILM—an extensible language modeling toolkit. In: Seventh international conference on spoken language processing. Denver, CO, pp 901–904
Tobias R (1995) An introduction to partial least squares regression. In: Proceedings of the twentieth annual sas users group international conference, Cary, NC: SAS Institute Inc. Orlando, FL, pp 1250–1257
Ueffing N, Ney H (2004) Bayes decision rules and confidence measures for statistical machine translation. In: Proceedings of EsTAL Espana for natural language processing. Alicante, Spain, pp 70–81, Springer
Ueffing N, Ney H (2005) Word-level confidence estimation for machine translation using phrase-based translation models. In: HLT/EMNLP 2005: Human language technology conference and conference on empirical methods in natural language processing, proceedings of the conference. Vancouver, BC, Canada, pp 763–770
Uhrik C, Ward W (1997) Confidence metrics based on N-Gram language model backoff behaviors. In: Fifth European conference on speech communication and technology. Rhodes, Greece, pp 2771–2774
Wold S, Ruhe A, Wold H, Dunn W III (1984) The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. SIAM J Sci Stat Comput 5(3):735–743
Zhang R, Rudnicky A (2001) Word level confidence annotation using combinations of features. In: Seventh European conference on speech communication and technology. Aalborg, Denmark, pp 2105–2108