FBK-UPV-UEdin participation in the WMT14 Quality Estimation shared-task

José G. C. de Souza[*]
University of Trento
Fondazione Bruno Kessler
Trento, Italy
desouza@fbk.eu

Jesús González-Rubio[*]
PRHLT Group
U. Politècnica de València
Valencia, Spain
jegonzalez@prhlt.upv.es

Christian Buck[*]
University of Edinburgh
School of Informatics
Edinburgh, Scotland, UK
cbuck@lantis.de

Marco Turchi, Matteo Negri
Fondazione Bruno Kessler
turchi,negri@fbk.eu

Abstract

This paper describes the joint submission of Fondazione Bruno Kessler, Universitat Politècnica de València and University of Edinburgh to the Quality Estimation tasks of the Workshop on Statistical Machine Translation 2014. We present our submissions for Task 1.2, 1.3 and 2. Our systems ranked first for Task 1.2 and for the Binary and Level1 settings in Task 2.

1 Introduction

Quality Estimation (QE) for Machine Translation (MT) is the task of evaluating the quality of the output of an MT system without reference translations. Within the WMT 2014 QE Shared Task four evaluation tasks were proposed, covering both word and sentence level QE. In this work we describe the Fondazione Bruno Kessler (FBK), Universitat Politècnica de València (UPV) and University of Edinburgh (UEdin) approach and system setup for the shared task.

We developed models for two sentence-level tasks: Task 1.2, scoring for post-editing effort, and Task 1.3, predicting post-editing time, and for all word-level variants of Task 2, binary and multiclass classification. As opposed to previous editions of the shared task, this year the participants were not supplied with the MT system that was used to produce the translation. Furthermore no system-internal features were provided. Thus, while the trained models are tuned to detect the errors of a specific system the features have to be generated independently (black-box).

2 Sentence Level QE

We submitted runs to two sentence-level tasks: Task 1.2 and Task 1.3. The first task aims at predicting the Human mediated Translation Edit Rate (HTER) (Snover et al., 2006) between a suggestion generated by a machine translation system and its manually post-edited version. The data set contains 1,104 English-Spanish sentence pairs post-edited by one translator (896 for training and 208 for test). The second task requires to predict the time, in milliseconds, that was required to post edit a translation given by a machine translation system. Participants are provided with 858 English-Spanish sentence pairs, source and suggestion, along with their respective post-edited sentence and post-editing time in seconds (650 data points for training and 208 for test). We participated in the scoring mode of both tasks.

2.1 Features

For our sentence-level submissions we compute features using different resources that do not use the MT system internals. We use the same set of features for both Task 1.2 and 1.3.

QuEst Black-box features (quest79). We extract 79 black-box features that capture the complexity, fluency and adequacy aspects of the QE problem. These features are extracted using the implementation provided by the QuEst framework (Specia et al., 2013). Among them are the 17 baseline features provided by the task organizers.

The **complexity** features are computed on the source sentence and indicate the complexity of translating the segment. Examples of these features are the language model (LM) probabilities of the source sentence computed in a corpus of the source language, different surface counts like the number of punctuation marks and the number of tokens in the source sentence, among others.

The **fluency** features are computed over the translation generated by the MT system and indicate how fluent the translation is in the target

[*]Contributed equally to this work.

language. One example would again be the LM probability of the translation given by a LM model trained on a corpus of the target language. Another example is the average number of occurrences of the target word within the target segment.

The third aspect covered by the QuEst features is the **adequacy** of the translation with respect to the source sentence, i.e., how the meaning of the source is preserved in the translation. Examples of features are the ratio of nouns, verbs and adjectives in the source and in the translation. For a more detailed description of the features in this group please refer to (Specia et al., 2013).

Word alignment (wla). Following our last year's submission (de Souza et al., 2013a) we explore information about word alignments to extract quantitative (amount and distribution of the alignments) and qualitative features (importance of the aligned terms). Our assumption is that features that explore what is aligned can bring improvements to tasks where sentence-level semantic relations need to be identified. We train the word alignment models with the MGIZA++ toolkit (Gao and Vogel, 2008) implementation of the IBM models (Brown et al., 1993). The models are built on the concatenation of Europarl, News Commentary, and MultiUN parallel corpora made available in the QE shared task of 2013, comprising about 12.8 million sentence pairs. A more detailed description of the 89 features extracted can be found in (de Souza et al., 2013a; de Souza et al., 2013b).

Word Posterior Probabilities (wpp). Using an external SMT system we produce 100k-best lists from which we derive Word Posterior Probabilities as detailed in Subsection 3.1.

We use the geometric mean of these probabilities to derive a sentence-level score.

Because the system that we use to produce the N-best list is not the same that generated the suggestions some suggested words never appear in the N-best list and thus receive zero probability. To overcome this issue we first clip the WPPs to a minimum probability. Using a small sample of the data to estimate this number we arrive at:

$$\log(p)_{min} = -2.$$

N-best diversity (div). Using the same 100k-best list as above we extract a number of measures that grasp the spatial distribution of hypotheses in the search space as described in (de Souza et al., 2013a).

Word Prediction (wpred). We introduce the use of the predictions provided by the word-level QE system described in Section 3 to leverage information for the sentence-level tasks. We combine the **binary** word-level predictions in different ways, with the objective of measuring the fluency of the translation in a more fine-grained way. We target a quantitative aspect of the words by computing ratios of `OK` or `BAD` predictions. Furthermore, we also explore a qualitative aspect by calculating ratios of different classes of words given by their part-of-speech tags, indicating the quality of distinct meaningful regions that compose the translation sentence. In total, we compute 18 features:

- number of `OK` predictions divided by the no. of words in the translation sentence (1 feature);
- number of `OK` function/content words predictions divided by the no. of function/content words in the translation (2 features);
- number of `OK` nouns, verbs, proper-nouns, adjective, pronouns predictions divided by the total nouns, verbs, proper-nouns, adjective, pronouns (5 features);
- size of the longest sequence of `OK`/`BAD` word predictions divided by the total number of `OK`/`BAD` predictions in the translation (2 features);
- number of `OK` predicted n-grams divided by the total number of n-grams in the translation. We vary n from 2 to 5 (4 features);
- number of words predicted as `OK` in the first/second half of the translation divided by the total number of words in the first/second half of the translation (2 features).
- number of words predicted as `OK` in the first/second quarter of the translation divided by the total number of words in the first/second quarter of the translation (2 features).

For some instances of the sentence-level tasks we were not able to produce word-level predictions due to an incomplete overlap between the word-level and sentence-level tasks datasets. For such data points we use the median of the feature column for Task 1.2 and the mean for Task 1.3.

Method	Features	Train T1.2	Train T1.3	Test T1.2	Test T1.3
SVR	baseline	16.90	16864	15.23	21490
ET	baseline	16.25	17888	17.73	19400
ET	quest79 + wla + wpp	15.62	17474	14.44	18658
ET	quest79 + wla + wpp + div^2	15.57	17471	14.38	18693
ET	quest79 + wla + wpp + div + wpred^1	15.05	16392	12.89	17477

Table 1: Training and test results for Task 1.2 and 1.3. Scores are the MAE on a development set randomly sampled from the training data (20%). Baseline features were provided by the shared task organizers. We used Support Vector Machines (SVM) regression to train the baseline models (first row). Submissions are marked with ^{1} and ^{2} for primary and secondary, respectively.

2.2 Experimental Setup

We build the sentence-level models for both tasks (T1.2 and T1.3) with the features described in Section 2.1 using one learning algorithm: extremely randomized trees (ET) (Geurts et al., 2006). ET is an ensemble of randomized trees in which each decision tree can be parameterized differently. When a tree is built, the node splitting step is done at random by picking the best split among a random subset of the input features. All the trees are grown on the whole training set and the results of the individual trees are combined by averaging their predictions. The models produced by this method demonstrated to be robust to a large number of input features. For our experiments and submissions we used the ET implementation included in the Scikit-learn library (Pedregosa et al., 2011).

During training we evaluate the models on a development set. The development set was obtained by randomly sampling 20% of the training data. The remaining 80% were used for training. The training process was carried out by optimizing the ET hyper-parameters with 100 iterations of random search optimization (Bergstra and Bengio, 2012) set to minimize the mean absolute error (MAE)[1] on 10-fold cross-validation over the training data. The ET hyper-parameters optimized are: the number of decision trees in the ensemble, the maximum number of features to consider when looking for the best split, the maximum depth of the trees used in the ensembles, the minimal number of samples required to split a node of the tree, and the minimum number of samples in newly created leaves. For the final submissions we run the random search with 1000 iterations over the whole training dataset.

[1]Given by $MAE = \frac{\sum_{i=1}^{N} |H(s_i) - V(s_i)|}{N}$, where $H(s_i)$ is the hypothesis score for the entry s_i and $V(s_i)$ is the gold standard value for s_i in a dataset with N entries.

2.3 Results

We train models on different combinations of feature groups (described in Section 2.1). Experiments results are summarized in Table 1. We have results with baseline features for both SVR and the ET models. For Task 1.2, adding features from different groups leads to increasing improvements. The combination of the *quest79*, *wla* and *wpp* groups outperforms the SVR baseline for Task 1.2 but not for Task 1.3. However, when compared to the ET model trained with the baseline features, it is possible to observe improvements with this group of features. In addition, adding the *div* group on top of the previous three leads to marginal improvements for both tasks. The best feature combination is given when adding the features based on the word-level predictions, configuring the combination of all the feature groups together (a total of 221 features). For both tasks this is our primary submission. The contrastive run for both tasks is the best feature group combination without the word-prediction-based features, *quest79*, *wla*, *wpp* and *div* for Task 1.2 and *quest79*, *wla*, *wpp* for Task 1.3.

Results on the test set can be found in the two last columns of Table 1 and are in line with what we found in the training phase. The rows that do not correspond to the official submissions and that are reported on the test set are experiments done after the evaluation phase. For both tasks the improvements increase as we add features on top of the baseline feature set and the best performance is reached when using the word prediction features with all the other features. The SVR baselines performance are the official numbers provided by the organizers. For Task 1.2 our primary submission achieves a MAE score lower than the score achieved during the training phase, showing that the model is robust. For Task 1.3, however, we do not observe such trend. Even though the primary submission for this task consistently improves over the other feature combinations, it does not outperform the score obtained during the training phase. This might be explained due to the difference in the distribution between training and test labels. In Task 1.2 the two distributions are more similar than in Task 1.3, which presents slightly different distributions between training and test data.

3 Word-Level QE

Task 2 is the word-level quality estimation of automatically translated news sentences without given reference translations. Participants are required to produce a label for each word in one or more of the following settings:

Binary classification: a `OK/BAD` label, where `BAD` indicates the need for editing the word.

Level1 classification: `OK`, `Accuracy`, or `Fluency` label specifying a coarser level of errors for each word, or `OK` for words with no error.

Multi-Class classification: one of the 20 error labels described in the shared-task description or `OK` for words with no error.

We submit word-level quality estimations for the English-Spanish translation direction. The corpus contains 1957 training sentences for a total of 47411 Spanish words, and 382 test sentences for a total of 9613 words.

3.1 Features

Word Posterior Probabilities (WPP) In order to generate an approximation of the decoder's search space as well as an N-best list of possible translations we re-translate the source using the system that is available for the 2013 WMT QE Shared Task (Bojar et al., 2013).

Certainly, there is a mismatch between the original system and the one that we used but, since our system was trained using the same news domain as the QE data, we assume that both face similar ambiguous words or possible reorderings. Using this system we generate a 100k-best list which is the foundation of several features.

We extract a set of word-level features based on posterior probabilities computed over N-best lists as proposed by previous works (Blatz et al., 2004; Ueffing and Ney, 2007; Sanchis et al., 2007).

Consider a target word e_i belonging to a translation $\mathbf{e} = e_1 \ldots e_i \ldots e_{|\mathbf{e}|}$ generated from a source sentence $\mathbf{f}$. Let $\mathcal{N}(\mathbf{f})$ be the list of N-best translations for $\mathbf{f}$. We compute features as the normalized sum of probabilities of those translations $\mathcal{S}(e_i) \subseteq \mathcal{N}(\mathbf{f})$ that "contain" word e_i:

$$\frac{1}{\sum_{\mathbf{e}'' \in \mathcal{N}(\mathbf{f})} \mathrm{P}(\mathbf{e}'' \mid \mathbf{f})} \sum_{\mathbf{e}' \in \mathcal{S}(e_i)} \mathrm{P}(\mathbf{e}' \mid \mathbf{f}) \quad (1)$$

where $\mathrm{P}(\mathbf{e} \mid \mathbf{f})$ is the probability translation $\mathbf{e}$ given source sentence $\mathbf{f}$ according to the SMT model.

We follow (Zens and Ney, 2006) and extract three different WPP features depending on the criteria chosen to compute $\mathcal{S}(e_i)$:

$\mathcal{S}(e_i) = \{\mathbf{e}' \in \mathcal{N}(\mathbf{f}) \mid \mathbf{a} = \mathrm{Le}(\mathbf{e}', \mathbf{e}) \wedge e'_{a_i} = e_i\}$
$\mathcal{S}(e_i)$ contain those translations $\mathbf{e}'$ for which the word Levenshtein-aligned (Levenshtein, 1966) to position i in $\mathbf{e}$ is equal to e_i.

$\mathcal{S}(e_i) = \{\mathbf{e}' \in \mathcal{N}(\mathbf{f}) \mid e'_i = e_i\}$
A second option is to select those translations $\mathbf{e}'$ that contain the word $\mathbf{e}_i$ at position i.

$\mathcal{S}(e_i) = \{\mathbf{e}' \in \mathcal{N}(\mathbf{f}) \mid \exists i' : e'_{i'} = e_i\}$
As third option, we select those translations $\mathbf{e}'$ that contain the word e_i, disregarding its position.

Confusion Networks (CN) We use the same N-best list used to compute the WPP features in the previous section to compute features based on the graph topology of confusion networks (Luong et al., 2014). First, we Levenshtein-align all translations in the N-best list using $\mathbf{e}$ as skeleton, and merge all of them into a confusion network. In this network, each word-edge is labelled with the posterior probability of the word. The output edges of each node define different *confusion sets* of words, each word belonging to one single confusion set. Each complete path passing through all nodes in the network represents one sentence in the N-best list, and must contain exactly one link from each confusion set. Looking to the confusion set which the hypothesis word belongs to, we extract four different features: maximum and minimum probability in the set (2 features), number of alternatives in the set (1 feature) and entropy of the alternatives in the set (1 feature).

Language Models (LM) As language model features we produced n-gram length/backoff behaviour and conditional probabilities for every word in the sentence. We employed both an interpolated LM taken from the MT system discussed in Section 3 as well as a very large LM which we built on 62 billion tokens of monolingual data extracted from Common Crawl, a public web crawl. While generally following the procedure of Buck et al. (2014) we apply an additional lowercasing step before training the model.

Word Lexicons (WL) We compute two different features based on statistical word lexicons (Blatz et al., 2004):

Avg. probability: $\frac{1}{|\mathbf{f}|+1} \sum_{j=0}^{|\mathbf{f}|} \mathrm{P}(e_i \mid f_j)$

Max. probability: $\max_{0 \le j \le |\mathbf{f}|} \mathrm{P}(e_i \mid f_j)$

where $\mathrm{P}(e \mid f)$ is a probabilistic lexicon, and f_0 is the source "NULL" word (Brown et al., 1993).

POS tags (POS) We extract the part-of-speech (POS) tags for both source and translation sentences using TreeTagger (Schmid, 1994). We use the actual POS tag of the target word as a feature. Specifically, we represent it as a *one-hot* indicator vector where all values are equal to zero except the one representing the current tag of the word, which is set to one. Regarding the source POS tags, we first compute the lexical probability of each target word given each source word. Then, we compute two different feature vectors for each target word. On the one hand, we use an indicator vector to represent the POS tag of the maximum probability source word. On the other hand, we sum up the indicator vectors for all the source words each one weighted by the lexical probability of the corresponding word. As a result, we obtain a vector that represents the probability distribution of source POS tags for each target word. Additionally, we extract a binary feature that indicates whether the word is a *stop word* or not.[2]

Stacking (S) Finally, we also exploit the diverse granularity of the word labels. The word classes for the Level1 and Multi-class conditions are fine grained versions of the Binary annotation, i.e. the `OK` examples are the same for all cases.

We re-use our binary predictions as an additional feature for the finer-grained classes. However, due to time constrains, we were not able to run the proper nested cross-validation but used a model trained on all available data, which therefore over-fits on the training data. Cross-validation results using the stacking approach are thus very optimistic.

3.2 Classifiers

We use bidirectional long short-term memory recurrent neural networks (BLSTM-RNNs) as implemented in the RNNLib package (Graves, 2008). Recurrent neural networks are a connectionist model containing a self-connected hidden layer. The recurrent connection provides information of previous inputs, hence, the network can benefit from past contextual information. Long short-term memory is an advanced RNN architecture that allows context information over long periods of time. Finally, BLSTM-RNNs combine bidirectional recurrent neural networks and the long short-term memory architecture allowing forward and backward context information. Using such context modelling classifier we can avoid the use of context-based features that have been shown to lead to only slight improvements in QE accuracy (González-Rubio et al., 2013).

As a secondary binary model we train a CRF. Our choice of implementation is Pocket CRF[3] which, while currently unmaintained, implements continuous valued features. We use a history of size 2 for all features and perform 10-fold cross-validation, training on 9 folds each time.

3.3 Experimental Setup

The free parameters of the BLSTM-RNNs are optimized by 10-fold cross-validation on the training set. Each cross-validation experiment consider eight folds for training, one held-out fold for development, and a final held-out fold for testing. We estimate the neural network with the eight training folds using the prediction performance in the validation fold as stopping criterion. The result of each complete cross-validation experiment is the average of the results for the predictions of the ten held-out test folds. Additionally, to avoid noise due to the random initialization of the network, we repeat each cross-validation experiment ten times and average the results. Once the optimal values of the free parameters are established, we estimate a new BLSTM-RNN using the full training corpus and we use it as the final model to predict the class labels of the test words.

Since our objective is to detect words that need to be edited, we use the weighted averaged F_1 score over the different class labels that denote an error as our main performance metric ($\mathrm{wF1}_{\mathrm{err}}$). We also report the weighted averaged F_1 scores

[2]`https://code.google.com/p/stop-words/`

[3]`http://pocket-crf-1.sourceforge.net/`

		Binary		Level1		MultiClass	
Method	**Features**	$\mathrm{wF1_{err}}$	$\mathrm{wF1_{all}}$	$\mathrm{wF1_{err}}$	$\mathrm{wF1_{all}}$	$\mathrm{wF1_{err}}$	$\mathrm{wF1_{all}}$
BLSTM-RNNs	LM+WPP+CN+WL	35.9	63.0	23.7	59.4	10.7	55.5
	+POS	**38.5**[1]	62.7	26.7[1]	59.5	12.7[1]	55.5
	+Stacking	—	—	**82.9**[2]	93.9	**64.7**[2]	88.0
CRF	LM+WPP+CN+WL+POS	39.5[2]	62.4 0	—	—	—	—

Table 2: Cross-validation results for the different setups tested for Task 2. Our two submissions are marked as ([1]) and ([2]) respectively.

over all the classes ($\mathrm{wF1_{all}}$).

3.4 Results

Table 2 presents the $\mathrm{wF1_{err}}$ and $\mathrm{wF1_{all}}$ scores for different sets of features. Our initial experiment includes language model (LM), word posterior probability (WPP), confusion network (CN), and word lexicon (WL) features for a total of 11 features. We extend this basic feature set with the indicator features based on POS tags for a total of 163 features. We further extend the feature vectors by adding the stacking feature in a total of 164 features.

Analyzing the results we observe that prediction accuracy is quite low. Our hypothesis is that this is due to the skewed class distribution. Even for the binary classification scenario (the most balanced of the three conditions), OK labels account for two thirds of the samples. This effect worsens with increasing number of error classes and the resulting sparsity of observations. As a result, the system tends to classify all samples as OK which leads to the low F_1 scores presented in Table 2.

We can observe that the use of POS tags indicator features clearly improved the prediction accuracy of the systems in the three conditions. This setup is our primary submission for the three conditions of task 2.

In addition, we observe that the use of the stacking feature provides a considerable improvement in prediction accuracy for Level1 and MultiClass. As discussed above the cross-validation results for the stacking features are very optimistic. Test predictions using this setup are our contrastive submission for Level1 and MultiClass conditions.

Results achieved on the official test set can be found in Table 3. Much in line with our cross-validation results the stacking-features prove helpful, albeit by a much lower margin. For the binary task the RNN model strongly outperforms the CRF.

Setup	**Binary**	**Level1**	**MultiClass**
BLSTM-RNN	**48.7**	37.2	17.1
+ Stacking	—	**38.5**	**23.1**
CRF	42.6	—	**—**

Table 3: Test results for Task 2. Numbers are weighted averaged F_1 scores (%) for all but the OK class.

4 Conclusion

This paper describes the approaches and system setups of FBK, UPV and UEdin in the WMT14 Quality Estimation shared-task. In the sentence-level QE tasks 1.2 (predicting post-edition effort) and 1.3 (predicting post-editing time, in ms) we explored different features and predicted with a supervised tree-based ensemble learning method. We were able to improve our results by exploring features based on the word-level predictions made by the system developed for Task 2. Our best system for Task 1.2 ranked first among all participants.

In the word-level QE task (Task 2), we explored different sets of features using a BLSTM-RNN as our classification model. Cross-validation results show that POS indicator features, despite sparse, were able to improve the results of the baseline features. Also, the use of the stacking feature provided a big leap in prediction accuracy. With this model, we ranked first in the Binary and Level1 settings of Task 2 in the evaluation campaign.

Acknowledgments

This work was supported by the MateCat and Casmacat projects, which are funded by the EC under the 7^{th} Framework Programme. The authors would like to thank Francisco Álvaro Muñoz for providing the RNN classification software.

References

James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proceedings of the international conference on Computational Linguistics*, pages 315–321.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19:263–311.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram Counts and Language Models from the Common Crawl. In *Proceedings of the Language Resources and Evaluation Conference*.

José G. C. de Souza, Christian Buck, Marco Turchi, and Matteo Negri. 2013a. FBK-UEdin participation to the WMT13 Quality Estimation shared-task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 352–358.

José G. C. de Souza, Miquel Esplá-Gomis, Marco Turchi, and Matteo Negri. 2013b. Exploiting qualitative information from automatic word alignment for cross-lingual nlp tasks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 771–776.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Jesús González-Rubio, José R. Navarro-Cerdan, and Francisco Casacuberta. 2013. Partial least squares for word confidence estimation in machine translation. In *6th Iberian Conference on Pattern Recognition and Image Analysis, (IbPRIA) LNCS 7887*, pages 500–508. Springer.

Alex Graves. 2008. Rnnlib: A recurrent neural network library for sequence learning problems. `http://sourceforge.net/projects/rnnl/`.

Vladimir Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Ngoc-Quang Luong, Laurent Besacier, and Benjamin Lecouteux. 2014. Word confidence estimation and its integration in sentence quality estimation for machine translation. In *Knowledge and Systems Engineering*, volume 244, pages 85–98. Springer.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn : Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Alberto Sanchis, Alfons Juan, and Enrique Vidal. 2007. Estimation of confidence measures for machine translation. In *Proceedings of the Machine Translation Summit XI*, pages 407–412.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12, pages 44–49.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Association for Machine Translation in the Americas*.

Lucia Specia, Kashif Shah, José G. C. de Souza, and Trevor Cohn. 2013. QuEst–a translation quality estimation framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 79–84.

Nicola Ueffing and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33:9–40.

Richard Zens and Hermann Ney. 2006. N-gram posterior probabilities for statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 72–77.