SHEF-NN: Translation Quality Estimation with Neural Networks

Kashif Shah[§], **Varvara Logacheva**[§], **Gustavo Henrique Paetzold**[§], **Frederic Blain**[§]
Daniel Beck[§], **Fethi Bougares**[†], **Lucia Specia**[§]
§Department of Computer Science, University of Sheffield, UK
{kashif.shah,v.logacheva,ghpaetzold1,f.blain,debeck1,l.specia}
@sheffield.ac.uk

†LIUM, University of Le Mans, France
fethi.bougares@lium.univ-lemans.fr

Abstract

We describe our systems for Tasks 1 and 2 of the WMT15 Shared Task on Quality Estimation. Our submissions use (i) a continuous space language model to extract additional features for Task 1 (SHEF-GP, SHEF-SVM), (ii) a continuous bag-of-words model to produce word embeddings as features for Task 2 (SHEF-W2V) and (iii) a combination of features produced by QuEst++ and a feature produced with word embedding models (SHEF-QuEst++). Our systems outperform the baseline as well as many other submissions. The results are especially encouraging for Task 2, where our best performing system (SHEF-W2V) only uses features learned in an unsupervised fashion.

1 Introduction

Quality Estimation (QE) aims at measuring the quality of the Machine Translation (MT) output without reference translations. Generally, QE is addressed with various features indicating fluency, adequacy and complexity of the source-translation text pair. Such features are then used along with Machine Learning methods in order for models to be learned.

Features play a key role in QE. A wide range of features from the source segments and their translations, often processed using external resources and tools, have been proposed. These go from simple, language-independent features, to advanced, linguistically motivated features. They include features that rely on information from the MT system that generated the translations, and features that are oblivious to the way translations were produced. This leads to a potential bottleneck: feature engineering can be time consuming, particularly because the impact of features vary across datasets and language pairs. Also, most features in the literature are extracted from segment pairs in isolation, ignoring contextual clues from other segments in the text. The focus of our contributions this year is to introduce a new set of features which are language-independent, require minimal resources, and can be extracted in unsupervised ways with the use of neural networks.

Word embeddings have shown their potential in modelling long distance dependencies in data, including syntactic and semantic information. For instance, neural network language models (Bengio et al., 2003) have been successfully explored in many problems including Automatic Speech Recognition (Schwenk and Gauvain, 2005; Schwenk, 2007) and Machine Translation (Schwenk, 2012). While neural network language models predict the next word given a preceding context, (Mikolov et al., 2013b) proposed a neural network framework to predict the word given the left and right contexts, or to predict the word's left and right contexts in a given sentence. Recently, it has been shown that these distributed vector representations (or word embeddings) can be exploited across languages to predict translations (Mikolov et al., 2013a). The word representations are learned from large monolingual data independently for source and target languages. A small seed dictionary is used to learn mapping from the source into the target space. In this paper, we investigate the use of such resources in both sentence-level (Task 1) and word-level QE (Task 2). As we describe in what follows, we extract features from such resources and use them to learn prediction models.

2 Continuous Space Language Model Features for QE

Neural networks model non-linear relationships between the input features and target outputs.

Proceedings of the Tenth Workshop on Statistical Machine Translation, pages 342–347,
Lisboa, Portugal, 17-18 September 2015.

They often outperform other techniques in complex machine learning tasks. The inputs to the neural network language model used here (called Continuous Space Language Model (CSLM)) are the h_j context words of the prediction: $h_j = w_{j-n+1}, ..., w_{j-2}, w_{j-1}$, and the outputs are the posterior probabilities of all words of the vocabulary: $P(w_j|h_j)\ \forall i \in [1, N]$ where N is the vocabulary size. CSLM encodes inputs using the so called one-hot coding, i.e., the ith word in the vocabulary is coded by setting all element to 0 except the ith element. Due to the large size of the output layer (vocabulary size), the computational complexity of a basic neural network language model is very high. Schwenk et al. (2012) proposed an implementation of the neural network with efficient algorithms to reduce the computational complexity and speed up the processing using a subset of the entire vocabulary called *short list*.

As compared to shallow neural networks, deep neural networks can use more hidden layers and have been shown to perform better. In all CSLM experiments described in this paper, we use deep neural networks with four hidden layers: a first layer for the word projection (320 units for each context word) and three hidden layers of 1024 units for the probability estimation. At the output layer, we use a $softmax$ activation function applied to a *short list* of the 32k most frequent words. The probabilities of the out of the *short list* words are obtained using a standard back-off n-gram language model. The training of the neural network is done by the standard back-propagation algorithm and outputs are the posterior probabilities. The parameters of the models are optimised on a held out development set.

Our CSLM models were trained with the CSLM toolkit [1]. We extracted the probabilities for Task 1's training, development and test sets for both source and its translation with their respective optimised models and used them as features along with other available features in a supervised learning algorithm. In Table 1, we report detailed statistics on the monolingual data used to train the back-off LM and CSLM. The training dataset consists of Europarl, News-commentary and News-crawl corpora with the Moore-Lewis data selection method (Moore and Lewis, 2010) to select the CSLM training data with respect to a Task's development set. The CSLM models are tuned using a concatenation of newstest2012 and newstest2013 of WMT's translation track.

Lang.	Train	Dev	LM px	CSLM px
en	4.3G	137.7k	164.63	116.58
es	21.2M	149.4k	145.49	87.14

Table 1: Training and dev datasets size (in number of tokens) and models perplexity (px).

3 Word Embedding Features for QE

The word embeddings used in our experiments are learned with the *word2vec* tool[2], introduced by (Mikolov et al., 2013b). The tool produces word embeddings using the Distributed Skip-Gram or Continuous Bag-of-Words (CBOW) models. The models are trained through the use of large amounts of monolingual data with a neural network architecture that aims at predicting the neighbours of a given word. Unlike standard neural network-based language models for predicting the next word given the context of preceding words, a CBOW model predicts the word in the middle given the representation of the surrounding words, while the Skip-Gram model learns word embedding representations that can be used to predict a word's context in the same sentence. As suggested by the authors, CBOW is faster and more adequate for larger datasets, so we used this model in our experiments.

We trained 500-dimensional representations with CBOW for all words in the vocabulary. We consider a 10-word context window to either side of the target word, sub-sampling option to 1e-05, and estimate the probability of a target word with the negative sampling method, drawing 10 samples from the noise distribution. The data used to train the models is the same as presented in Table 1. We then extracted word embeddings for all words in the Task 2 training, development and test sets from these models to be used as features. These distributed numerical representations of words as features aim at locating each word as a point in a 500-dimensional space.

Inspired by the work of (Mikolov et al., 2013a), we extracted another feature by mapping the source space onto a target space using a seed dictionary (trained with Europarl + News-commentary + News-crawl). A given word and

[1] http://www-lium.univ-lemans.fr/cslm/

[2] https://code.google.com/p/word2vec/

its continuous vector representation a could be mapped to the other language space by computing $z = Ma$, where M is a transformation matrix learned with stochastic gradient descent. The assumption is that the vector representations of similar words in different languages are related by a linear transformation because of similar geometric arrangements. The words whose representation are closest to a in the target language space, using cosine similarity, are considered as potential translations for a given word in the source language. Since the goal of QE is not to translate content, but to measure the quality of translations, we take the source-to-target similarity scores as a feature itself. To calculate it, we first learn word alignments (see Section 4.2.2), and then compute the similarity scores between target word and the source word aligned to it.

4 Experiments

We present experiments on the WMT15 QE Tasks 1 and 2, with CSLM features for Task 1, and word embedding features for Task 2.

4.1 Task 1

4.1.1 Dataset

Task 1's English-Spanish dataset consists respectively of a training set and development set with $11,271$ and $1,000$ source segments, their machine translations, the post-editions of the latter, and edit distance scores between between the MT and its post-edited version (HTER). The test set consists of $1,817$ English-Spanish source-MT pairs. Translations are produced by a single online statistical MT system. Each of the translations was post-edited by crowdsourced translators, and HTER labels were computed using the TER tool (settings: tokenised, case insensitive, exact matching only, with scores capped to 1).

4.1.2 Feature set

We extracted the following features:

- **AF**: 80 black-box features using the QuEst framework (Specia et al., 2013; Shah et al., 2013a) as described in Shah et al. (2013b).
- **CSLM**: A feature for each source and target sentence using CSLM as described in Section 2.
- **FS(AF)**: Top 20 features selected from the above 82 features with Gaussian Processes (GPs) by the procedure described in (Shah et al., 2013b).

4.1.3 Learning algorithms

We use the Support Vector Machines implementation in the `scikit-learn` toolkit (Pedregosa et al., 2011) to perform regression (SVR) on each feature set with either linear or RBF kernels and parameters optimised using grid search.

We also apply GPs with similar settings to those in our WMT13 submission (Beck et al., 2013) using `GPy` toolkit [3]. For models with feature selection, we train a GP, select the top 20 features according to the produced feature ranking, and then retrain a SparseGP on the full training set using these 20 features and 50 inducing points. To evaluate the prediction models we use Mean Absolute Error (MAE), its squared version – Root Mean Squared Error (RMSE), and Spearman's Correlation.

4.2 Task 2

4.2.1 Dataset

The data for this is the same as the one provided in Task 1. All segments have been automatically annotated for errors with binary word-level labels ("GOOD" and "BAD") by using the alignments provided by the TER tool (settings: tokenised, case insensitive, exact matching only, disabling shifts by using the '-d 0' option) between machine translations and their post-edited versions. The edit operations considered as errors ("BAD") are replacements and insertions.

4.2.2 Word alignment training

To extract word embedding features, as explained in Section 3, we need word-to-word alignments between source and target data. As word-level alignments between the source and target corpora were not made available by WMT, we first aligned the QE datasets with a bilingual word-level alignment model trained on the same data used for the *word2vec* modelling step, with the help of the GIZA++ toolkit (Och and Ney, 2003). Working on target side, we refined the resulting n-m target-to-source word alignments to a set of 1-m alignments by filtering potential spurious source-side candidates out. To do so, the decision was based on the lexical probabilities extracted from the previous alignment training step. Hence, each target-

[3] `http://sheffieldml.github.io/GPy/`

side token has been aligned to the source-side candidate with the highest lexical probability. To map our two monolingual vector spaces trained with word embedding models, we extracted a bilingual dictionary with the same settings used for word-alignment.

4.2.3 Data filtering

An inspection of the training and development data showed that 15% of the sentences contain no errors and are therefore less useful for model learning. In addition, most sentences have very low HTER score, showing that very few words are considered incorrect. Figure 1 shows the HTER scores distribution for the training dataset: 50% of the sentences have the HTER of 0.15 or lower (points below the bottom orange line on the Figure), 75% of the sentences have the score of 0.28 or lower (points below the middle green line). The distributions for the development and test sets are similar.

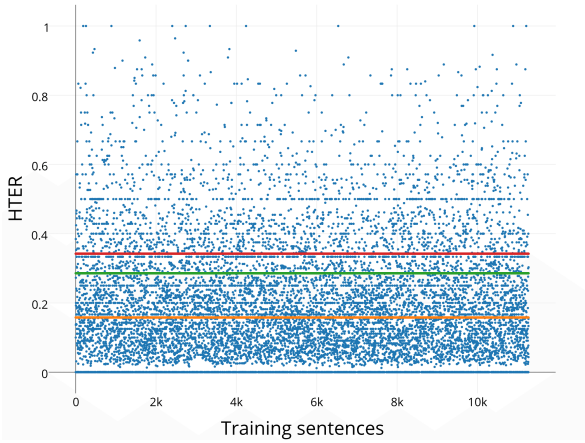

Figure 1: The distribution of HTER scores for the training data. Below orange line – 50% of the data, below green line – 75% of the data, above red line – worst 2000 sentences (18% of the data).

Sentences with few or no edits lead to models that tag more words as "GOOD", so the tagging is too optimistic, resulting in higher F1 score for the "GOOD" class but lower F1 score for the "BAD" class. This is an issue as obtaining a good F1 score for the "BAD" class is arguably the primary goal of a QE model (and also the main evaluation criterion for the task). Therefore, we decided to increase the percentage of "BAD" labels in the training data by filtering out sentences which have zero or too few errors. As a filtering strategy, we took only sentences with the highest proportions of editing.

We performed experiments with two subsets of the training sentences with the highest HTER score: 2,000 samples (18% of the data, i.e., points above the top red line in Figure 1); and 5,000 samples (44% of the data). Since the F1-score for the "BAD" class was higher on the dev set for the model built from the smaller subset, we chose it to perform the tagging for the official submission of the shared task. This subset contains sentences with HTER score from 0.34 to 1, an average score of 0.49, and variance of 0.018.

4.2.4 Learning algorithms

We learned binary tagging models for both SHEF-W2V and SHEF-QuEst++ using a Conditional Random Fields (CRF) algorithm (Lafferty et al., 2001). We used **pystruct** (Müller and Behnke, 2014) for SHEF-W2V, and **CRFSuite** (Okazaki, 2007) for SHEF-QuEst++. Both tools allow one to train a range of models. For pystruct we used the linear-chain CRF trained with a structured SVM solver, which is the default setting. For CRFSuite we used the Adaptive Regularization of Weight Vector (AROW) and Passive Aggressive (PA) algorithms, which have been shown to perform well in the task (Specia et al., 2015).

Systems are evaluated in terms of classification performance (Precision, Recall, F1) against the "GOOD" and "BAD" labels, and their weighted average of both F1 scores (W-F1). The main evaluation metric is the average F1 score for the "BAD" label.

4.3 Results

4.3.1 Task 1

We trained various models with different feature sets and algorithms and evaluated the performance of these models on the official development set. The results are shown in Table 2. Some interesting findings:

- SVM performed better than GP.
- SVM with linear kernel performed better than with RBF kernel.
- CSLM features alone performed better than the baseline features.
- CSLM features always bring improvements whenever added to either baseline or complete feature set.

System.	Kernel	Features	#. of Feats.	MAE	RMSE	Spear. Corr
Baseline (SVM)	RBF	BL	17	0.1479	0.1965	0.1651
SHEF-SVM	RBF	CSLM	2	0.1474	0.1959	0.1911
SHEF-SVM	RBF	BL+CSLM	19	0.1464	0.1950	0.1924
SHEF-SVM	RBF	AF	80	0.1497	0.1944	0.2259
SHEF-SVM	RBF	AF+CSLM	82	0.1452	0.1920	0.2325
SHEF-SVM	Linear	AF+CSLM	82	0.1422	0.1889	0.2736
SHEF-SVM	Linear	AF(FS)	20	0.1459	0.1896	0.2465
SHEF-GP	RBF	AF(FS)	20	0.1493	0.1917	0.2187

Table 2: Results on development set of Task 1.

System.	MAE	RMSE	DeltaAvg	Spear. Corr
Baseline	0.15	0.19	0.22	0.13
SHEF-SVM	0.14	0.18	0.51	0.28
SHEF-GP	0.15	0.19	0.31	0.28

Table 3: Official results on test set of Task 1.

- Linear SVM with selected features by GP achieves comparable results to linear SVM with the full feature set (82).
- Both CSLM features appear in the top 20 selected features by GP.

Based on these findings, as official submissions for Task 1, we put forward a system with linear SVM using 82 features, and another with GP on the selected feature set. The official results are shown in Table 3.

4.3.2 Task 2

For the SHEF-QuEst++ system, we combined all 40 features described in (Specia et al., 2015) with the source-to-target similarity feature described in Section 3. For the SHEF-W2V system, we tried several settings on the development data in order to define the best-performing set of features and dataset size. We used two feature sets:

- 500-dimensional word embedding vectors for the target word only.
- 500-dimensional word embedding vectors for the target word and the source word aligned to it.

In addition, both these feature sets included the source-to-target similarity feature. We performed the data filtering technique described in 4.2.3, and tested the systems using:

- The full dataset.
- 5K sentences with the highest HTER score.
- 2K sentences with the highest HTER score.

System	W-F1	F1 Bad	F1 Good
Baseline	75.48	17.07	89.07
MONO-ALL	72.31	0.35	89.39
MONO-5000	74.47	14.82	88.63
MONO-2000	65.83	35.38	73.06
MONO-2000-SIM	65.87	35.53	73.07
BI-ALL	72.23	0.0	89.38
BI-5000	75.37	22.77	87.86
BI-2000	64.78	38.64	70.99
BI-2000-SIM	64.56	38.45	70.76
QuEst++-AROW-SIM	68.58	38.54	75.72
QuEst++-PA-SIM	26.42	34.86	24.42

Table 4: Results on development set of Task 2.

Results on the development set are outlined in Table 4. The system names are formed as follows: "MONO" or "BI" indicate that the SHEF-W2V system was trained on the target or target+source word embeddings feature set. "ALL", "5000" and "2000" indicate that we used the entire training set, $5,000$ sentences or $2,000$ sentences, respectively. The prefix "SIM" means that the feature sets were enhanced with the vector similarity feature. Finally, "AROW" and "PA" correspond to the two learning algorithms used by SHEF-QuEst++.

Combining the target and source-side word embedding vectors was found to improve the performance of SHEF-W2V compared to using only target-side vectors. The impact of the similarity feature is less clear: it slightly improved the performance of the monolingual feature set, but decreased the scores for the bilingual feature set. We can also notice that the AROW algorithm is much more effective than the PA algorithm for SHEF-QuEst++.

Filtering out sentences that are mostly correct allowed to achieve much higher F1-scores for the "BAD" class. The best results were achieved with a relatively small subset of the data (18%). Therefore, as our official submissions, we chose the model using bilingual vectors trained on 2,000 sentences with the highest HTER score, and the same model extended with the similarity feature.

System.	W-F1	F1 Bad	F1 Good
Baseline	75.71	16.78	88.93
W2V-BI	65.73	38.43	71.63
W2V-Bi-SIM	65.27	38.40	71.52
QuEst++-AROW	64.69	37.69	71.11
QuEst++-AROW-SIM	62.07	38.36	67.58
QuEst++-PA	33.02	35.16	32.51
QuEst++-PA-SIM	26.25	34.30	24.38

Table 5: Official results on test set of Task 2.

The results on the test set are presented in Table 5, in which it is shown that the source-to-target similarity feature has gain 0.67% in F1 of "BAD" labels for SHEF-QuEst++ system with the AROW algorithm.

5 Conclusions

We have proposed several novel features for translation quality estimation, which are trained with the use of neural networks. When added to large standard feature sets for Task 1, the CSLM features led to improvements in prediction. Moreover, CSLM features alone performed better than baseline features on the development set. Combining the source-to-target similarity feature with the ones produced by QuEst++ improved its performance in terms of F1 for the "BAD" class. Finally, the results obtained by SHEF-W2V are quite promising: although it uses only features learned in an unsupervised fashion, it was able to outperform the baseline as well as many other systems.

Acknowledgements

This work was supported by the QT21 (H2020 No. 645452, Lucia Specia, Frederic Blain), Cracker (H2020 No. 645357, Kashif Shah) and EXPERT (EU FP7 Marie Curie ITN No. 317471, Varvara Logacheva) projects and funding from CNPq/Brazil (No. 237999/2012-9, Daniel Beck).

References

Daniel Beck, Kashif Shah, Trevor Cohn, and Lucia Specia. 2013. Shef-lite: When less is more for translation quality estimation. *Proceedings of WMT13*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th ICML*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL 2013*.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the 48th ACL*.

Andreas C. Müller and Sven Behnke. 2014. pystruct - learning structured prediction in python. *Journal of Machine Learning Research*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields. `http://www.chokkan.org/software/crfsuite/`.

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*.

Holger Schwenk and Jean-Luc Gauvain. 2005. Training neural network language models on very large corpora. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of COLING*.

Kashif Shah, Eleftherios Avramidis, Ergun Biçicic, and Lucia Specia. 2013a. Quest - design, implementation and extensions of a framework for machine translation quality estimation. *Prague Bull. Math. Linguistics*.

Kashif Shah, Trevor Cohn, and Lucia Specia. 2013b. An investigation on the effectiveness of features for translation quality estimation. In *Proceedings of the Machine Translation Summit*.

Lucia Specia, Kashif Shah, José G. C. de Souza, and Trevor Cohn. 2013. QuEst - A translation quality estimation framework. In *Proceedings of 51st ACL*.

Lucia Specia, Gustavo H. Paetzold, and Carolina Scarton. 2015. Multi-level translation quality prediction with quest++. In *Proceedings of The 53rd ACL*.