Black Box Features for the WMT 2012 Quality Estimation Shared Task

Christian Buck
School of Informatics
University of Edinburgh
Edinburgh, UK, EH8 9AB
christian.buck@ed.ac.uk

Abstract

In this paper we introduce a number of new features for quality estimation in machine translation that were developed for the WMT 2012 quality estimation shared task. We find that very simple features such as indicators of certain characters are able to outperform complex features that aim to model the connection between two languages.

1 Introduction and Task

This paper describes the features and setup used in our submission to the WMT 2012 quality estimation (QE) shared task. Given a machine translation (MT) system and a corpus of its translations which have been rated by humans, the task is to build a predictor that can accurately estimate the quality of further translations. The human ratings range from 1 (incomprehensible) to 5 (perfect translation) and are given as the mean rating of three different judges.

Formally we are presented with a source sentence f_1^J and a translation e_1^I and we need to assign a score $S(f_1^J, e_1^I) \in [1, 5]$ or, in the ranking task, order the source-translation pairs by expected quality.

2 Resources

The organizers have made available a baseline QE system that consists of a number of well established features (Blatz et al., 2004) and serves as a starting point for development. Furthermore the MT system that generated the translations is available along with its training data. Compared to the large training corpus of the MT engine, the QE system is based on a much smaller training set as detailed in Table 1.

	# sentences
europarl-nc	1,714,385
train	1,832
test	422

Table 1: Corpus statistics

3 Features

In the literature (Blatz et al., 2004) a large number of features have been considered for confidence estimation. These can be grouped into four general categories:

1. *Source features* make a statement about the source sentence, assessing the difficulty of translating a particular sentence with the system at hand. Some sentences may be very easy to translate, e.g. short and common phrases, while long and complex sentences are still beyond the system's capabilities.

2. *Translation features* model the connection between source and target. While this is very closely related to the general problem of machine translation, the advantage in confidence estimation is that we can exercise unconstructive criticism, i.e. point out errors without offering a better translation. In addition, there is no need for an efficient search algorithm, thus allowing for more complex models.

3. *Target features* judge the translation of the system without regarding in which way it was produced. They often resemble the language

Proceedings of the 7th Workshop on Statistical Machine Translation, pages 91–95,
Montréal, Canada, June 7-8, 2012.

model used in the noisy channel formulation (Brown et al., 1993) but can also pinpoint more specific issues. In practice, the same features as for the source side can be used; the interpretation however is different.

4. *Engine features* are often referred to as *glass box features* (Specia et al., 2009). They describe the process which produced the translation in question and usually rely on the inner workings of the MT system. Examples include model scores and word posterior probabilities (WPP) (Ueffing et al., 2003).

In this work we focus on the first three categories and ignore the particular system that produced the translations. Such features are commonly referred to as *black box features*. While some glass box features, e.g. word posterior probabilities, have led to promising results in the past, we chose to explore new features potentially applicable to translations from any source, e.g. translations found on the web.

3.1 Binary Indicators

MTranslatability (Bernth and Gdaniec, 2001) gives a notion of the structural complexity of a sentence that relates to the quality of the produced translation. In the literature, several characteristics that may hinder proper translation have been identified, among them poor grammar and misplaced punctuation. As a very simple approximation we implement binary indicators that detect clauses by looking for quotation marks, hyphens, commas, etc. Another binary feature marks numbers and uppercase words.

3.2 Named Entities

Another aspect that might pose a potential problem to MT is the occurrence of words that were only observed a few times or in very particular contexts, as it is often the case for Named Entities. We used the Stanford NER Tagger (Finkel et al., 2005) to detect words that belong to one of four groups: Person, Location, Organization and Misc. Each group is represented by a binary feature.

Counts are given in Table 2. The test set has significantly less support for the *Misc* category, possibly hinting that this data was taken from a different source or document. To avoid the danger of biasing the classifier we decided not to use the *Misc* indicator in our experiments.

	train (src)		test (src)	
	abs	rel	abs	rel
Person	623	34%	141	33%
Location	479	26%	99	23%
Organization	505	28%	110	26%
Misc	428	23%	53	13%

Table 2: Distribution of Named Entities. The counts are based on a binary features, i.e. multiple occurrences are treated as a single one.

3.3 Backoff Behavior

In related work (Raybaud et al., 2011) the backoff behavior of a 3-gram LM was found to be the most powerful feature for word level QE. We compute for each word the longest seen n-gram (up to $n = 4$) and take the average length as a feature. N-grams at the beginning of a sentence are extended with <s> tokens to avoid penalizing short sentences. This is done on both the source and target side.

3.4 Discriminative Word Lexicon

Following the approach of Mauser et al. (2009) we train log-linear binary classifiers that directly model $p(e|f_1^J)$ for each word $e \in e_1^I$:

$$p(e|f_1^J) = \frac{exp\left(\sum_{f \in f_1^J} \lambda_{e,f}\right)}{1 + exp\left(\sum_{f \in f_1^J} \lambda_{e,f}\right)} \quad (1)$$

where $\lambda_{e,f}$ are the trained model weights. Please note that this introduces a global dependence on the source sentence so that every source word may influence the choice of all words in e_1^I as opposed to the local dependencies found in the underlying phrase-based MT system.

Assuming independence among the words in the translated sentence we could compute the probability of the sentence pair as:

$$p(e_1^I|f_1^J) = \prod_{e \in e_1^I} p(e|f_1^J) \cdot \prod_{e \notin e_1^I} \left(1 - p(e|f_1^J)\right). \quad (2)$$

In practice the second part of Equation (2) is too noisy to be useful given the large number of words

source	~~resumption~~ of the session
target	reanudación del período de ~~sesiones~~

Table 3: Example entry of filtered training corpus.

that do not appear in the sentence at hand. We therefore focus on the observed words and use the geometric mean of their individual probabilities:

$$x_{\mathrm{DWL}}(f_1^J, e_1^I) = \left(\prod_{e \in e_1^I} p(e|f_1^J) \right)^{1/I} . \quad (3)$$

We also compute the probability of the lowest scoring word as an additional feature:

$$x_{\mathrm{DWLmin}}(f_1^J, e_1^I) = \min_{e \in e_1^I} p(e|f_1^J). \quad (4)$$

3.5 Neural Networks

We seek to directly predict the words in e_1^I using a neural network. In order to do so, both source and target sentence are encoded as high dimensional vectors in which positive entries mark the occurrence of words. This representation is commonly referred to as the *vector space model* and has been successfully used for information retrieval.

The dimension of the vector representation is determined by the respective sizes of the source and target vocabulary. Without further pre-processing we would need to learn a mapping from a 90k ($|V_f|$) to a 170k ($|V_e|$) dimensional space. Even though our implementation is specifically tailored to exploit the sparsity of the data, such high dimensionality makes training prohibitively expensive.

Two approaches to reduce dimensionality are explored in this work. First, we simply remove all words that never occur in the QE data of 2,254 sentences from the corpus leaving 8,365 input and 9,000 output nodes. This reduces the estimated training time from 11 days to less than 6 hours per iteration[1]. Standard stochastic gradient decent on a three-layer feed-forward network is used.

As shown in Table 3 the filtering can lead to artifacts in which case an erroneous mapping is learned. Moreover the filtering approach does not scale well as the QE corpus and thereby the vocabulary grows.

[1] using a 2.66 GHz Intel Xeon and 2 threads

Our second approach to reduce dimensionality uses the *hashing trick* (Weinberger et al., 2009): a hash function is applied to each word and the sentence is represented by the hashed values which are again transformed using vector space model as above. The dimensionality reduction is due to the fact that there are less possible hash values than words in the vocabulary. To reduce the loss of information due to collisions, several different hash functions are used. The resulting vector representation closely resembles a Bloom Filter (Bloom, 1970).

This approach scales well but introduces two new parameters: the number of hash functions to use and the dimensionality of the resulting space. In our experiments we have used SHA-1 hashes with three different salts of which we used the first 12 bits, thereby mapping the sentences into a 4096-dimensional space.

The results presented in Section 4 based on networks with 500 hidden nodes which were trained for at least 10 iterations. The networks are not trained until convergence due to time constraints; additional training iterations will likely result in better performance. Experiments using 250 or 1000 hidden nodes showed very similar results.

After the models are trained we compare the predicted and the observed target vectors and derive two features: (i) the euclidean distance, denoted as NNdist and HNNdist for the filtered and hashed versions respectively and (ii) the geometric mean of those dimensions where we expect a positive value, denoted as NNprop+ and HNNprob+ in Table 5.

3.6 Edit Distance

Using Levenshtein Distance we computed the distance to the closest entry in the training corpus. The idea is that a sentence that was already seen almost identically would be easier to translate. Likewise, a translation that is very close to an element of the corpus is likely to be a good translation. This was performed for both source and target side and on character as well as on word level giving a total of four (EDIT) scores. The scores are normalized by the length of the respective lines.

source corpus	“	”	"
europarl-nc	37	227	25,637
train	0	0	641
test	78	76	100

Table 4: Counts of different quotation mark characters.

4 Experiments

In this work we focus on the prediction of human assessment of translation quality, i.e. the regression task of the WMT12 QE shared task. Our submission for the ranking task is derived from the order implied by the predicted scores without further re-ranking.

In general our efforts were directed towards feature engineering and not to the machine learning aspects. Therefore, we apply a standard pipeline and use neural networks for regression. All parameter tuning is performed using 5-fold cross validation on the baseline set of 17 features as provided by the organizers.

4.1 Preprocessing and Analysis

To avoid including our own judgment, no more than the first ten lines of the test data were visually inspected in order to ensure that the training and test data was preprocessed in the same manner. Furthermore, the distribution of individual characters was investigated. As shown in Table 4, the test data differs from the training corpus in treatment of quotation marks. Hence, we replaced all typographical quotation marks (“, ”) with the standard double quote symbol (").

Prior to computation of the features described in Subsections 3.3, 3.4 and 3.5 all numbers are replaced with a special `$number` token.

Baseline features are used without further scaling; experiments where all features were scaled to the $[0, 1]$ range showed a drop in accuracy.

While we implemented the training ourselves for the features presented in Subsection 3.5, the open source neural network library FANN[2] is used for all experiments in this section. As the performance of individual classifiers shows a high variance, presumably due to local minima, all experiments are conducted using ensembles on 500 networks trained with random initialization. Their consensus is computed as the average of the individual predictions.

[2]http://leenissen.dk/fann/wp/

Feature (Section)	MAE	RMSE	|PCC|
BACKOFF (3.3)	0.0	0.0	
INDICATORS (3.1)	+0.5	+0.7	
NER (3.2)	+0.5	+0.4	
DWLmin (3.4)	−0.1	−0.1	0.19
DWL (3.4)	0.0	−0.1	0.36
EDIT (3.6) - tgt words	0.0	0.0	0.32
EDIT (3.6) - tgt chars	−0.1	0.0	0.27
EDIT (3.6) - src words	0.0	0.0	0.36
EDIT (3.6) - src chars	+0.2	+0.1	0.37
NNdist (3.5)	0.0	0.0	0.35
NNprob+ (3.5)	+0.1	+0.2	0.35
HNNdist (3.5)	0.0	0.0	0.37
HNNprob+ (3.5)	+0.1	+0.1	0.35

Table 5: Analysis of individual features using 5-fold cross-validation. Positive values indicate improvement over a baseline of MAE 57.7% and RMSE 72.7%; e.g. including the DWL feature actually worsens RMSE from 72.7% to 72.8%.
The last column gives the Pearson correlation coefficient between the feature and the score if the feature is a single column. This information was not used in feature selection as it is not based on cross validation.

4.2 Feature Evaluation

To evaluate the contribution of individual features, each feature is tested in conjunction with all baseline features, using the parameters that were optimized on the baseline set. This slightly favors the baseline features but we still expect that expressive additional features lead to a noticeable performance gain. The results are detailed in Table 5. In addition to the main evaluation metrics, mean average error (MAE) and root mean squared error (RMSE), we report the Pearson correlation coefficient (PCC) as a measure of predictive strength of a single feature. Because features are not used alone this does not directly translate into overall performance. Still, it can be observed that our proposed features show good correlation to the target variable. For comparison, among the baseline features only 2 of 17 reach a PCC of over 0.3.

While the results generally remain inconclusive, some very simple features that indicate difficulties

for the translation engine show good performance. In particular binary markers of named entities and and the indicator features introduced in Subsection 3.1 perform well. Further experiments with the latter show their contribution to the systems performance can be attributed to a single feature: the indicator of the genitive case, i.e. occurrences of **'s** or **s'**.

Testing more combinations of simple and complex features may lead to improvements at the risk of over-fitting on the cross validation setup. As a simple remedy several feature sets were created at random, always combining all baseline features and several new features presented in this paper. Averaging of the individual results of all sets that performed better than the baseline resulted in our submission.

4.3 Results and Discussion

Of all the features detailed only a few lead to a considerable improvement. This is also reflected by our results on the test data which are nearly indistinguishable from the performance of the baseline system. While this is disappointing, our more complex features introduce a number of free parameters and further experimentation will be needed to conclusively assess their usefulness. In particular, features based on neural networks can be further optimized and tested in other settings.

Even though the machine learning aspects of this task are not the focus of this work we are confident that the proposed setup is sound and can be reused in further evaluations.

5 Conclusion

We described a number of new features that can be used to predict human judgment of translation quality. Results suggest pointing out sentences that are hard to translate, e.g. because they are too complex, is a promising approach.

We presented a detailed evaluation of the utility of individual features and a solid baseline setup for further experimentation. The system, based on an ensemble of neural networks, is insensitive to parameter settings and yields competitive results.

Our new features can potentially be applied for a multitude of applications and may deliver insights into the fundamental problems that cause translation errors, thus aiding the progress in MT research.

Acknowledgments

This work was supported by the MateCAT project, which is funded by the EC under the 7^{th} Framework Programme.

References

Arendse Bernth and Claudia Gdaniec. 2001. Mtranslatability. *Machine Translation*, 16(3):175–218, September.

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 Coling 2004*, pages 315–321, Geneva, Switzerland, August.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL 2005, pages 363–370, Stroudsburg, PA, USA.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Conference on Empirical Methods in Natural Language Processing*, pages 210–217, Singapore, August.

Sylvain Raybaud, David Langlois, and Kamel Smaïli. 2011. "this sentence is wrong." detecting errors in machine-translated sentences. *Machine Translation*, 25(1):1–34, March.

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. 2009. Estimating the sentence-level quality of machine translation systems. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, EAMT-2009, pages 28–35, Barcelona, Spain, May.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. 2003. Confidence measures for statistical machine translation. In *Machine Translation Summit*, pages 394–401, New Orleans, LA, September.

Kilian Q. Weinberger, Anirban Dasgupta, John Langford, Alexander J. Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, ACM International Conference Proceeding Series, pages 1113–1120, Montreal, Quebec, Canada, June.