Efforts on Machine Learning over Human-mediated Translation Edit Rate

Eleftherios Avramidis
German Research Center for Artificial Intelligence (DFKI)
Language Technology Lab
Alt Moabit 91c, 10559 Berlin, Germany
eleftherios.avramidis@dfki.de

Abstract

In this paper we describe experiments on predicting HTER, as part of our submission in the Shared Task on Quality Estimation, in the frame of the 9th Workshop on Statistical Machine Translation. In our experiment we check whether it is possible to achieve better HTER prediction by training four individual regression models for each one of the edit types (deletions, insertions, substitutions, shifts), however no improvements were yielded. We also had no improvements when investigating the possibility of adding more data from other non-minimally post-edited and freely translated datasets. Best HTER prediction was achieved by adding deduplicated WMT13 data and additional features such as (a) rule-based language corrections (language tool) (b) PCFG parsing statistics and count of tree labels (c) position statistics of parsing labels (d) position statistics of tri-grams with low probability.

1 Introduction

As Machine Translation (MT) gets integrated into regular translation workflows, its use as base for post-editing is radically increased. As a result, there is a great demand for methods that can automatically assess the MT outcome and ensure that it is useful for the translator and can lead to more productive translation work.

Although many agree that the quality of the MT output itself is not adequate for the professional standards, there has not yet been a widely-accepted way to measure its quality on par with human translations. One such metric, the Human Translation Edit Rate (HTER) (Snover et al., 2006), is the focus of the current submission. HTER is highly relevant to the need of adapting MT to the needs of translators, as it aims to measure how far it is from an acceptable equivalent translation done by humans.

HTER is used here in the frame of Quality Estimation, i.e. having the goal of being able to predict the post-editing effort in a real case environment, right before the translation is given to the user, without real access to the correct translation. For this purpose the text of the source and the produced translation is analyzed by automatic tools in order to infer indications (numerical features) that may be relevant to the quality of the translation. These features are used in a statistical model whose parameters are estimated with common supervised Machine Learning techniques.

This work presents an extensive search over various set-ups and parameters for such techniques, aiming to build a model that better predicts HTER over the data of the Shared Task of the 9th Workshop on Statistical Machine Translation.

2 New approaches being tested

2.1 Break HTER apart

HTER is a complex metric, in the sense that it is calculated as a linear function over specific types of *edit distance*. The official algorithm performs a comparison between the MT output and the corrected version of this output by a human translator, who performed the minimum number of changes. The comparison results in counting the number of insertions, deletions, substitutions and shifts (e.g. reordering). The final HTER score is the total number of edits divided by the number of reference words.

$$\text{HTER} = \frac{\#\text{insertions} + \#\text{dels} + \#\text{subs} + \#\text{shifts}}{\#\text{reference words}}$$

We notice that the metric is clearly based on four edit types that are seemingly independent of each other. This poses the question whether the existing approach of learning the entire metric altogether introduces way too much complexity in the machine learning process. Instead, we test the hypothesis that it is more effective to build a separate model for each error type and then put the output of each model on the overall HTER fraction shown above.

Following this idea, we score the given translations again in order to produce all four HTER factors (insertions, deletions, substitutions and shifts) and we train four regression models accordingly. This way, each model can be optimized separately, in order to better fit the particular error type, unaffected by the noise that other error types may infer.

2.2 Rounding of individual edit type predictions

Due to the separate model per error type, it is possible to perform corrections on the predicted error count for each error type, before the calculation of the entire HTER score. This may be helpful, given the observation that continuous statistical models may produce a real number as prediction for the count of edits, whereas the actual requirement is an integer.

Here, we take this opportunity and test the hypothesis that prediction of the overall HTER is better, if the output of the four individual models is rounded to the closest integer, before entered in the HTER ratio.

2.3 More data by approximating minimal post-edits

We investigate whether prediction performance can be improved by adding further data. This rises from the fact that the original number of sentences is relatively small, given the amount of usable features. Unfortunately, the amount of openly available resources of minimally post-edited translations are few, given the fact that this relies on a costly manual process usually done by professionals.

Consequently, we add more training samples, using reference translations of the source which are not post-edited. In order to ensure that the additional data still resemble minimally post-edited translations as required for HTER, we include those additional sentences only if they match specific similarity criteria. In particular, the translations are filtered, based on the amount of edits between the MT output and the reference translation; sentences with an amount of edits above the threshold are omitted.

3 Methods

3.1 Machine Learning on a regression problem

Fitting a statistical model in order to predict continuous values is clearly a regression problem. The task takes place on a sentence level, given a set of features describing the source and translation text, and the respective edit score for the particular sentence.

For this purpose we use Support Vector Regression - SVR (Basak et al., 2007), which uses linear learning machines in order to map a non-linear function into a feature space induce by a high-dimensional kernel. Similar to the baseline, the RBF kernel was used, whose parameters where adjusted via a grid search, cross-validated (10 folds) on all data that was available for each variation of the training.

3.2 Features

As explained, the statistical model predicts the edit counts based on a set of features. Our analysis focuses on "black-box" features, which only look superficially on the given text and the produced translation, without further knowledge on how this translation was produced. These features depend on several automatic extraction mechanisms, mostly based on existing language processing tools.

3.2.1 Baseline features

A big set of features is adopted from the baseline of the Shared Task description:

Language models: provide the smoothed n-gram probability and the n-gram perplexity of the sentence.

Source frequency: A set of eight features includes the percentage of uni-grams, bi-grams and tri-grams of the processed sentence in frequency quartiles 1 (lower frequency words) and 4 (higher frequency words) in the source side of a parallel corpus (Callison-Burch et al., 2012).

Count-based features include count and percentage of tokens, unknown words, punctuation marks, numbers, tokens which do or do not contain characters "a-z"; the absolute difference between number of tokens in source and target normalized by source length, number of occurrences of the target word within the target hypothesis averaged for all words in the hypothesis (type/token ratio).

3.2.2 Additional features

Additionally to the baseline features, the following feature groups are considered:

Rule-based language correction is a result of hand-written controlled language rules, that indicate mistakes on several pre-defined error categories (Naber, 2003). We include the number of errors of each category as a feature.

Parsing Features: We parse the text with a PCFG grammar (Petrov et al., 2006) and we derive the counts of all node labels (e.g. count of verb phrases, noun phrases etc.), the parse log-likelihood and the number of the n-best parse trees generated (Avramidis et al., 2011). In order to reduce unnecessary noise, in some experiments we separate a group of "basic" parsing labels, which include only verb phrases, noun phrases, adjectives and subordinate clauses.

Position statistics: This are derivatives of the previous feature categories and focus on the position of unknown words, or node tree tags. For each of them, we calculate the average position index over the sentence and the standard deviation of these indices.

3.3 Evaluation

All specific model parameters were tested with cross validation with 10 equal folds on the training data. Cross validation is useful as it reduces the possibility of overfitting, yet using the entire amount of data.

The regression task is evaluated in terms of Mean Average Error (MAE).

4 Experiment setup

4.1 Implementation

The open source *language tool*[1] is used to annotate source and target sentences with automatically detected monolingual error tags. Language model features are computed with the SRILM toolkit (Stolcke, 2002) with an order of 5, based on monolingual training material from Europarl v7.0 (Koehn, 2005) and News Commentary (Callison-Burch et al., 2011). For the parsing parsing features we used the Berkeley Parser (Petrov and Klein, 2007) trained over an English and a Spanish treebank (Taulé et al., 2008).[2] Baseline features are extracted using Quest and HTER edits and scores are recalculated by modifying the original TERp code. The annotation process is organised with the Ruffus library (Goodstadt, 2010) and the learning algorithms are executed using the Scikit Learn toolkit (Pedregosa et al., 2011).

[1]Open source at http://languagetool.org

datasets	feature set	MAE
wmt14	baseline	0.142
wmt14	all features	0.143
wmt14,wmt13	baseline	0.140
wmt14,wmt13	all features	0.138

Table 1: Better scores are achieved when training with both WMT14 and deduplicated WMT13 data

4.2 Data

In our effort to reproduce HTER in a higher granularity, we noticed that HTER scoring on the official data was reversed: the calculation was performed by using the MT output as reference and the human post-edition as hypothesis. Therefore, the denominator on the "official" scores is the number of tokens on the MT output. This makes the prediction even easier, as this number of tokens is always known.

Apart from the data provided by the WMT14, we include additional minimally post-edited data from WMT13. It was observed that about 30% of the WMT13 data already occurred in the WMT14 set. Since this would negatively affect the credibility of the cross-fold evaluation (section 3.3) and also create duplicates, we filtered out incoming sentences with a string match higher than 85% to the existing ones.

The rest of the additional data (section 2.3) was extracted from the test-sets of shared tasks WMT2008-2011.

5 Results

5.1 Adding data from previous year

Adding deduplicated data from the HTER prediction task of WMT13 (Section 4.2) leads to an improvement of about 0.004 of MAE for the best feature-set, as it can be seen by comparing the respective entries of the two horizontal blocks of Table 1.

[2]although the Spanish grammar performed purely in this case and was eliminated as a feature

feature set	MAE
baseline (b)	0.140
b + language tool	0.141
b + source parse	0.141
b + parse pos	0.142
b + basic parse pos	0.139
b + parse count	0.139
b + low prob trigram pos	0.139
all without char count	0.139
all without lang. tool	0.139
all features	**0.138**

Table 2: Comparing models built with several different feature sets, including various combinations of the features described in section 3.2. All models trained on combination of WMT14 and WMT13 data

5.2 Feature sets

We tested separately several feature sets, additionally to the baseline feature set and the feature set containing all features. The feature sets tested are based on the feature categories explained in Section 3.2.2 and the results are seen in Table 2. One can see that there is little improvement on the MAE score, which is achieved best by using all features.

Adding individual categories of features on the baseline has little effect. Namely, the language tool annotation, the source parse features and the source and target parse positional features deteriorate the MAE score, when added to the baseline features.

On the contrary, there is a small positive contribution by using the position statistics of only the "basic" parsing nodes (i.e. noun phrases, verb phrases, adjectives and subordinate clauses). Similarly positive is the effect of the count of parsed node labels for source and target and the features indicating the position of tri-grams with low probability (lower than the deviation of the mean). Although language tool features deteriorate the score of the baseline model when added, their absense has a negative effect when compared to the full feature set.

5.3 Separate vs. single HTER predictor

Table 3 includes comparisons of models that test the hypothesis mentioned in Section 2.1. For both models trained over the baseline or with additional features, the MAE score is higher (worse), when

features	mode		MAE	std +/-
baseline	single		0.140	0.012
baseline	combined		0.148	0.018
baseline	combined	round	0.152	0.018
all	single		0.138	0.009
all	combined		0.160	0.019
all	combined	round	0.162	0.020

Table 3: The combination of 4 different estimators (combined) does not bring any improvement, when compared to the single HTER estimator. Models trained on both WMT14 and WMT13 data

separate models are trained. This indicates that our hypothesis does not hold, at least for the current setting of learning method and feature sets. Rounding up individual edit type predictions to the closes integer, before the calculation of the HTER ratio, deteriorates the scores even more.

5.4 Effect of adding non-postedited sentences

In Table 4 we can see that adding more data, which are not minimally post-edited (but normal references), does not contribute to a better model, even if we limit the number of edits. The lowest MAE is 0.176, when compared to the one of our best model which is 0.138.

The best score when additional sentences are imported, is achieved by allowing sentences that have between up to edits, and particularly up to 3 substitutions and up to 1 deletion. Increasing the number of edits on more than 4, leads to a further deterioration of the model. One can also see that adding training instances where MT outputs did not require any edit, also yields scores worse than the baseline.

6 Conclusion and further work

In our submission, we process the test set with the model using all features (Table 2). We additionally submit the model trained with additional filtered sentences, as indicated in the second row of Table 4.

One of the basic hypothesis of this experiment, that each edit type can better be learned individually, was not confirmed given these data and settings. Further work could include more focus on the individual models and more elaborating on features that may be specific for each error type.

del	ins	sub	shifts	total	add. sentences	MAE	std+/-
0	0	0	0	0	275	0.177	0.049
1	**0**	**3**	**0**	**4**	**480**	**0.176**	**0.040**
1	0	2	0	3	433	0.177	0.040
0	0	4	0	4	432	0.177	0.040
2	1	0	0	3	296	0.177	0.048
2	0	3	0	5	530	0.178	0.038
4	0	2	0	6	485	0.178	0.041
4	4	0	0	8	310	0.178	0.046
2	1	0	1	4	309	0.178	0.047
1	0	5	0	6	558	0.179	0.039
1	4	5	0	10	1019	0.200	0.031

Table 4: Indicative MAE scores achieved by adding filtered not minimally post-edited WMT translation

Acknowledgment

This work was supported from European Unions Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 296347 (QTLaunchPad).

References

Eleftherios Avramidis, Maja Popović, David Vilar, and Aljoscha Burchardt. 2011. Evaluate with Confidence Estimation : Machine ranking of translation outputs using grammatical features. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 65–70, Edinburgh, Scotland, July. Association for Computational Linguistics.

Debasish Basak, Srimanta Pal, and Dipak Chandra Patranabis. 2007. Support vector regression. *Neural Information Processing-Letters and Reviews*, 11(10):203–224.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 Workshop on Statistical Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland, July. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June. Association for Computational Linguistics.

Leo Goodstadt. 2010. Ruffus: a lightweight Python library for computational pipelines. *Bioinformatics*, 26(21):2778–2779.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. *Proceedings of the tenth Machine Translation Summit*, 5:79–86.

Daniel Naber. 2003. A rule-based style and grammar checker. Technical report, Bielefeld University, Bielefeld, Germany.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the 2007 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, New York. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Matthew Snover, B Dorr, Richard Schwartz, L Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Andreas Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 901–904. ISCA, September.

Mariona Taulé, Antònia Martí, and Marta Recasens. 2008. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA).