MIPT System for World-Level Quality Estimation

Mikhail Mosyagin
MIPT, Russia
mosyagin.md@phystech.edu

Varvara Logacheva
Neural Networks and Deep Learning Lab
MIPT, Russia
logacheva.vk@mipt.ru

Abstract

We explore different model architectures for the WMT 19 shared task on word-level quality estimation of automatic translation. We start with a model similar to Shef-bRNN (Ive et al., 2018), which we modify by using conditional random fields (CRFs) (Lafferty et al., 2001) for sequence labelling. Additionally, we use a different approach for labelling gaps and source words. We further develop this model by including features from different sources such as BERT (Devlin et al., 2018), baseline features for the task (Specia et al., 2018) and transformer encoders (Vaswani et al., 2017). We evaluate the performance of our models on the English-German dataset for the corresponding task.

1 Introduction

Current methods of assessing the quality of machine translation, like BLEU (Papineni et al., 2002), are based on comparing the output of a machine translation system with several gold reference translations. The tasks of quality estimation at the WMT 19 conference aims at detecting errors in automatic translation without a reference translation at various levels (word-level, sentence-level and document-level). In this work we predict word-level quality.

In the task the participants are given a source sentence and its automatic translation and are asked to label the words in the machine translation as *OK* or *BAD*. The machine translation system could have omitted some words in the translated sentence. To detect such errors participants are also asked to label the gaps in the automatic translation. A target sentence has a gap between every pair of neighboring words, one gap in the beginning of the sentence and one gap at the end of the sentence. We are also interested in detecting the words in the source sentence that led to errors in the translation. For this purpose participants are also asked to label the words in source sentences. The source labels were obtained based on the alignments between the source and the post-edited target sentences. If a target token is labeled as *BAD* in the translation, then all source tokens aligned to it are labeled as *BAD* as well.

In section 2 we introduce our base model, which is a modified version of phrase-level Shef-bRNN (Ive et al., 2018), and further develop it by using different methods of extracting features from the input alongside the bi-RNN features. In section 3 we write about our experimental setup and in section 4 we present the scores achieved by our models. In section 5 we summarize our work and propose ways for further development.

2 Models

All of our models have two stages: feature extraction and tag prediction. The first stage uses different neural architectures like bi-LSTM encoder and BERT (Devlin et al., 2018) to extract features from the input sequences. Some models also use human-crafted features alongside the automatically generated ones. The second stage feeds the sequence of extracted features into a CRF (Lafferty et al., 2001) to obtain labels for words or gaps in the automatic translation.

2.1 RNN Features

Our base model is similar to phrase-level Shef-bRNN (Ive et al., 2018). We chose the phrase-level version of Shef-bRNN over the word-level version because we found it to be more understandable and intuitive.

The model is given a sequence of source tokens $s_1, \ldots, s_n$ and a sequence of target tokens $t_1, \ldots, t_m$. The source sequence is fed into the source encoder, which is a bidirectional LSTM.

Proceedings of the Fourth Conference on Machine Translation (WMT), Volume 3: Shared Task Papers (Day 2) pages 92–96
Florence, Italy, August 1-2, 2019.

Thus, for every word s_j in the source a source vector $h_j^{\text{src}} = \left[\vec{h}_j^{\text{src}}, \overleftarrow{h}_j^{\text{src}}\right]$ is produced, where $\vec{h}_j^{\text{src}}$ and $\overleftarrow{h}_j^{\text{src}}$ are the corresponding hidden states of the forward and backward LSTMs and $[x, y]$ is the concatenation of vectors x and y. Similarly, the target sequence is fed into the target encoder, which is also a bidirectional LSTM, to obtain a target vector h_j^{tgt} for every word t_j in the target sequence. Global attention (Luong et al., 2015) is used to obtain context vector c_j for every target vector h_j^{tgt}:

$$\alpha_{ij} = h_i^{\text{src}\top} h_j^{\text{tgt}},$$

$$a_{ij} = \frac{\exp(\alpha_{ij})}{\sum_{k=1^n} \exp(\alpha_{kj})},$$

$$c_j = \sum_{k=1}^{n} a_{kj} h_k^{\text{src}}.$$

The vector c_j gives a summary of the source sentence, focusing on parts which are most relevant to the target token. Using the same technique, we obtain self-context vector sc_j for every target vector h_j^{tgt} by computing global attention for h_j^{tgt} over $h_i^{\text{tgt}}, i \neq j$. The resulting feature vector is denoted as $f_j^{\text{RNN}} = \left[h_j^{\text{tgt}}, c_j, sc_j\right]$ for every word t_j in the target sequence.

2.2 Baseline Features

Specia et al. (2018) use a CRF (Lafferty et al., 2001) with a set of human-crafted features as the baseline model for the same task at WMT 18. The WMT 18 and WMT 19 tasks use the same English-German dataset, so we can use the baseline features provided with the WMT 18 dataset to further improve the performance our model.

For every word t_j in the target sequence baseline features represent a sequence of 34 values: $b_j^1, \ldots, b_j^{34}$, some of which are numerical – like the word count in source and target sentences – and the others are categorical – like the target token, aligned source token and their part-of-speech (POS) tags. We represent categorical features using one-hot encoding. In this case if a value of a categorical feature occurs less than min_occurs times in the train dataset, then this value is ignored (i.e. it is represented by a zero vector). After the conversion all features are concatenated into a single feature vector f_j^{Base}

2.3 BERT Features

BERT is a model for language representation presented by Delvin et al. (2018) which demonstrated state of the art performance on several NLP tasks. BERT is trained on a word prediction task and, as shown in (Kim et al., 2017), word prediction can be helpful for the quality estimation task. Pretrained versions of BERT are publicly available and we use one of them to generate features for our models.

To extract BERT features the target sequence is fed into a pretrained BERT model. It is important to note that we do not fine-tune BERT and just use its pretrained version as-is. BERT utilizes WordPiece tokenization (Wu et al., 2016), so for each target token t_j it produces k_j output vectors $\text{BERT}_j^1, \ldots, \text{BERT}_j^{k_j}$. However, we can only use a fixed size feature vector for each source token. We noticed that about 83% of target tokens produce less than three BERT tokens. This means that by using only two of the produced tokens we will preserve most of the information. To obtain the BERT feature vector, we decided to concatenate the first and the last BERT outputs $f_j^{\text{BERT}} = \left[\text{BERT}_j^1, \text{BERT}_j^{k_j}\right]$. We chose the first and the last outputs, because this approach was the easiest to implement.

2.4 Transformer Encoder

We tried replacing bi-RNN encoders with transformer encoders (Vaswani et al., 2017) to include more contextual information in the encoder outputs.

The source transformer encoder produces embeddings $h_1^{\text{src}}, \ldots, h_n^{\text{src}}$ for the source sequence and the target transformer encoder produces outputs $h_1^{\text{tgt}}, \ldots, h_m^{\text{tgt}}$ for the target sequence. After that, similarly to 2.1, a context vector c_j is obtained for every word in the target sequence. For transformer encoder we do not compute self-context vectors as the transformer architecture itself utilizes the self-attention mechanism.

The resulting feature vector is denoted as $f_j^{\text{Trf}} = \left[h_j^{\text{tgt}}, c_j\right]$.

2.5 Word Labelling

After the feature vectors for the target sequence have been obtained, they are fed into a CRF that labels the words in the translation. In this paper we explore architectures that use the following feature vectors:

- RNN: f_j^{RNN};
- RNN+Baseline:

$$f_j^{\text{RNN+Baseline}} = \left[f_j^{\text{RNN}}, f_j^{\text{Base}}\right];$$

- RNN+BERT:

$$f_j^{\text{RNN+BERT}} = \left[f_j^{\text{RNN}}, f_j^{\text{BERT}}\right];$$

- RNN+Baseline+BERT:

$$f_j^{\text{RNN+Baseline+BERT}} = \\ = \left[f_j^{\text{RNN}}, f_j^{\text{Base}}, f_j^{\text{BERT}}\right];$$

- Transformer: f_j^{Trf}.
- Transformer+Baseline+BERT:

$$f_j^{\text{Transformer+Baseline+BERT}} = \\ = \left[f_j^{\text{Trf}}, f_j^{\text{Base}}, f_j^{\text{BERT}}\right];$$

To label words in the source sequence we use the alignments between the source sentence and the machine translation provided with the dataset. Specifically, if a source word s_j is aligned with a target word t_j, which is labeled as *BAD* then we label s_j as *BAD* as well. In case when s_j is aligned with multiple target words, we label s_j as *BAD* if at least one of the aligned target words is labeled *BAD*.

2.6 Gap Labelling

Unlike word-level Shef-bRNN, we refrain from using a dummy word to predict gap tags, because increasing the input sequence length might make it difficult for encoders to carry information between distant words. Instead, we train different models for word labelling and gap labelling.

To modify a word labelling architecture *Arch*, where *Arch* is either *RNN+Baseline+BERT* or *Transformer+Baseline+BERT*, to label gaps, we construct a new sequence of features:

$$fg_j^{\text{Arch}} = \left[f_j^{\text{Arch}}, f_{j+1}^{\text{Arch}}\right]$$

for $j = 0, \ldots, m$. Here we assume f_0^{Arch} and f_{m+1}^{Arch} to be zero vectors.

After the new sequence has been constructed, we feed it into a CRF to label the gaps.

3 Experimental Setup

We train and evaluate our models on the WMT 19 Word-Level Quality Estimation Task English-German dataset. In our experiments we did not utilize pre-training or multi-task learning unlike some versions of Shef-bRNN. All our models were implemented in PyTorch, the code is available online. [1]

For RNN feature extraction we use OpenNMT (Klein et al., 2017) bi-LSTM encoder implementation with 300 hidden units in both backward and forward LSTMs for models that label words and 150 hidden units for models that label gaps. We used FastText models (Grave et al., 2018) for English and German languages to produce word embeddings.

Baseline features were provided with the dataset. In our experiments we used $\text{min\_occurs} = 4$ when building baseline feature vocabularies.

Pretrained BERT model was provided by the *pytorch-pretrained-bert* package. [2] In our experiments we used the *bert-base-multilingual-cased* version of BERT.

We used the OpenNMT (Klein et al., 2017) transformer encoder implementation with the following parameters: $\text{num\_layers} = 3$, $\text{d\_model} = 300$, $\text{heads} = 4$, $\text{d\_ff} = 600$ (or $\text{d\_ff} = 300$ for gap labelling), $\text{dropout} = 0.1$.

We trained our models using PyTorch implementation of the ADADELTA algorithm (Zeiler, 2012) with all parameters, except the learning rate, set to their default values. For the train loss to converge we used the learning rate of 1 for the *RNN* and *Transformer* models, the learning rate of 0.3 for the *RNN+Baseline* model and the learning rate of 0.1 for *RNN+Bert*, *RNN+Baseline+Bert* and *Transformer+Baseline+Bert* models. The inputs were fed into the model in mini-batches of 10 samples.

4 Results

We used the English-German dataset provided in the WMT 19 Shared task on Word-Level Quality Estimation. The primary metric for each type of tokens – source words, target words and gaps – is

[1] https://github.com/Mogby/QualityEstimation

[2] https://github.com/huggingface/pytorch-pretrained-BERT

F_1 Mult which is the product of F_1 scores for *BAD* and *OK* labels.

The scores for each system are presented in Table 1 (participation results), Table 2 (target words), Table 3 (source words) and Table 4 (gaps). For the WMT 19 task we submitted the *RNN+Baseline+BERT* and *Transformer+Baseline+BERT* models which correspond to the *Neural CRF RNN* and *Neural CRF Transformer* entries in the public leaderboard.

We don't have the scores for the WMT 18 Baseline system and the Shef-bRNN system on the development dataset, so we can compare them directly with only two of our systems from table 1. Both of these systems perform on par with Shef-bRNN and the *Transformer+Baseline+BERT* model was able to achieve a slightly better score for target classification. Word-level Shef-bRNN seems to outperform all of our other systems, most likely, because it uses a more appropriate architecture for the task. All of our systems, seem to outperform the WMT 18 baseline system.

The BERT features turned out to improve the performance a little – an increase of 0.02 for target labelling and an increase of 0.01 for source labelling. The baseline features, on the other hand, have a greater impact on the model's performance, increasing the score by 0.05 for target labelling and by 0.04 for source labelling. Replacing the bi-RNN encoder with a transformer encoder also improved the score by 0.03 in case of the *RNN+Baseline+BERT* configuration.

5 Conclusion

We applied different neural systems to the task of word-level quality estimation. We measured their performance in comparison to each other and the baseline system for the task. All of our systems outperformed the WMT 18 baseline on the development dataset and can be trained in a couple of hours on a single Tesla K80 GPU.

Our models can be further improved by fine-tuning BERT and utilizing multi-task learning as proposed in (Kim et al., 2017).

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Julia Ive, Carolina Scarton, Frédéric Blain, and Lucia Specia. 2018. Sheffield submissions for the WMT18 quality estimation shared task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 794–800.

Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. pages 562–568.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

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 Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón Astudillo, and André F. T. Martins. 2018. Findings of the wmt 2018 shared task on quality estimation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 689–709, Belgium, Brussels. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason

Model	Target F_1 Mult	Source F_1 Mult
RNN + Baseline + BERT	0.30	0.26
Transformer + Baseline + BERT	**0.33**	**0.27**

Table 1: Final results on the test dataset.

Model	Dataset	F_1 OK	F_1 BAD	F_1 Mult
WMT 18 Baseline	test	0.20	0.92	0.18
Shef-bRNN (Word-Level)	test	0.86	0.35	0.30
RNN	dev	0.88	0.26	0.23
RNN + Baseline	dev	0.91	0.31	0.28
RNN + BERT	dev	0.88	0.29	0.25
RNN + Baseline + BERT	dev	0.88	0.34	0.30
Transformer	dev	0.90	0.25	0.23
Transformer + Baseline + BERT	dev	0.89	0.37	**0.33**

Table 2: Models scores on WMT 19 English-German dataset, target prediction. The baseline scores are taken from (Specia et al., 2018) and the Shef-bRNN scores are taken from (Ive et al., 2018)

Model	Dataset	F_1 OK	F_1 BAD	F_1 Mult
Shef-bRNN (Word-Level)	test	0.87	0.33	0.29
RNN	dev	0.88	0.22	0.19
RNN + Baseline	dev	0.90	0.25	0.23
RNN + BERT	dev	0.87	0.23	0.20
RNN + Baseline + BERT	dev	0.88	0.29	**0.25**
Transformer	dev	0.88	0.23	0.20
Transformer + Baseline + BERT	dev	0.89	0.28	**0.25**

Table 3: Models scores on WMT 19 English-German dataset, source prediction. The Shef-bRNN scores are taken from (Ive et al., 2018)

Model	Dataset	F_1 OK	F_1 BAD	F_1 Mult
Shef-bRNN (Word-Level)	test	0.99	0.12	0.12
RNN + Baseline + BERT	dev	0.98	0.14	0.14
Transformer + Baseline + BERT	dev	0.99	0.14	**0.14**

Table 4: Models scores on WMT 19 English-German dataset, gap prediction. The Shef-bRNN scores are taken from (Ive et al., 2018)

Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.