Sentence-level QE task 2020

Ivan Mang   Toma Popov   Guy Leroy

Abstract

This document contains our experiments, findings and results for a regression model estimating the quality of translations from English to Mandarin Chinese. A Jupyter notebook containing our different models is hosted on GitHub: https://github.com/gml16/sentence-level-quality-estimation.

1 Architecture & Design

To build our model, we chose PyTorch (Paszke et al., 2019) as it is easy to learn, dynamic and very flexible, working well with other libraries such as Numpy.

We began our investigation by exploring how good a score we could achieve with approaches such as SVM (Cortes and Vapnik, 1995) and Random Tree Forests (Ho, 1995). With SVM, we explored the different kernels as well as varying the degree for the polynomial kernel. Additionally, we tried varying the regularisation constant during to see if this made any significant difference. These approaches did not vary much in performance, with the best one achieving approximately 0.92 RMSE and 0.24 Pearson score (Benesty et al., 2009).

When trying Random Tree Forests, we found that using $n = 1000$ trees took a long time to run but still produced a poor score, so we decided that this approach was not sufficient to learn the function required to solve the task.

We then moved onto neural approaches, by first starting with a Feed-Forward NN (Sanger, 1989).

Since Feed-Forward NNs did not prove sufficient to model this problem, we tried more complex approaches in RNNs. One of our approaches was to use GRU layers (Cho et al., 2014), combined with a fully connected layer at the end. We chose GRU over vanilla RNN layers as this is better for preventing the short-term memory problem that can occur with RNNs. As sentences can become quite long, we thought that GRU layers should perform better than vanilla RNN as they should be able to better pick out the important information from the input. One of our main ideas was to use two GRU layers, one per language, so that the two sentence representations can be fed into separate RNN layers to be learned, and then the two can be linked together by the fully connected layer which could attempt to produce the final score. This architecture is illustrated in figure 1. Furthermore, we also tried a similar architecture to the GRU RNN but with LSTM layers instead (Hochreiter and Schmidhuber, 1997). This gave a very slight decrease in Pearson score over the GRU architecture and the number of epochs to converge was higher. However, the RMSE was essentially no different.

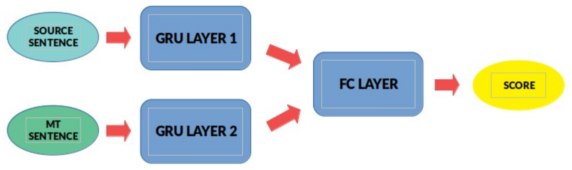

Figure 1: High-level forward steps of our RNN with GRU cells

For the loss function for our models, we arrived at the decision to simply use MSE, however, we did try out several of our own functions too. We investigated making our own loss functions which took into consideration the Pearson score, MSE and MAE, by computing a weighted sum of the three. We explored various weighting for these, including prioritising Pearson score entirely, however, none performed better than the simple MSE loss function.

We manually searched for better hyper-parameters, by trying out a range of values for each, to get an idea of what values may be best. Even though this approach is not as efficient as grid searching, it gave us better insights on how each hyper parameter influences our network. For the number of epochs, we explored values: 10, 50, 100, 150, 200. We found that past 100 epochs the network was over fitting so we chose this as our final value to use. For the batch size, we tried sizes: 32, 64, 128, 256 and found that 64 was optimal.

2 Pre-processing

The data we used to train our model covers general topics and is well written. Therefore, we decided to do light-weight text pre-processing.

For English sentences, we first lower case every words in the sentences. We tokenise the sentence into words then remove all stop words. We restricts string to alphabetic characters only and remove punctuation and numbers in the text.

For Chinese sentences, we use a library called Jieba to tokenise the sentences into segments. Just as we did with English sentences, we only allow tokens that are either Chinese or English words and removed punctuation and numbers. We included English tokens in the data because we did not want to remove too much information when a Chinese sentence contained English words.

3 Word embedding

We used pre-trained word embedding for both English and Chinese. For English, we used GloVe, which is a an unsupervised learning algorithm for obtaining vector representations for words, to turn the words tokens into vectors.

For Chinese, we used a ChineseT CoNLL17 corpus from Language Technology Group at the University of Oslo. We tried to use different corpus such as Tencent AI Lab Embedding Corpus for Chinese. However, we got a similar accuracy using them. We transformed the words into vectors of dimension 100, and removed words not included in the corpus.

We tried different approaches to handle the input data for the model. First, we averaged the vectors on each sentences and obtained a list of averaged sentences where each item was of dimension (1) [2].

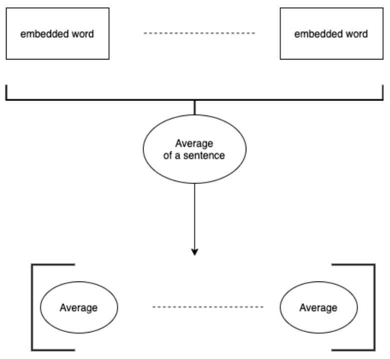

Figure 2: Text to vector using sentence embedding

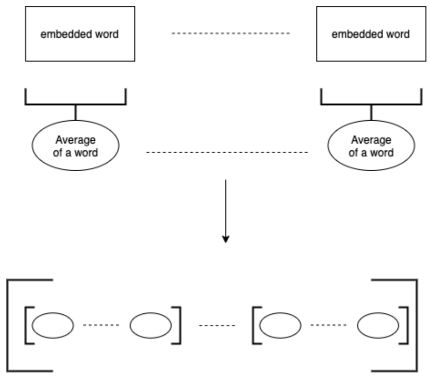

Figure 3: Second approach

This approach provided a model a list of float numbers, which are the averaged vector of the sentences. This did not perform well in our model and we want contain more information in the model. Therefore, we average each word vectors instead of the whole sentence [3]. The dimension of each sentence became (`word_dimension`).

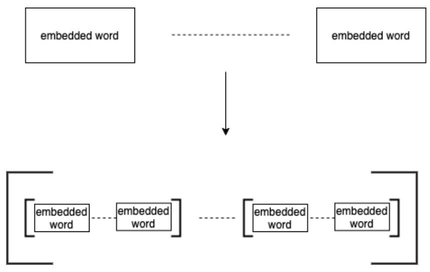

Figure 4: Third approach

It improved our accuracy a bit. On the later stage of the project, we decided to use RNN as our architecture, we did not need to average the words and reduce the dimension of our input. The input to the model could be the embedded words themselves[4]. Hence, the dimension of the input was now (`number_of_words`, `word_dimension`).

4 Performance

4.1 English to German

All the scores mentioned are when evaluating on the validation set. Starting from the baseline model, its best Pearson score was 0.062, using a poly SVM. We first made a simple multi layer perceptron and the Pearson score slightly improved to 0.08. This might have been a lucky epoch as the Pearson score oscillated between positive and negative numbers. We switched to the English to Chinese data sets as we had made more progress on the data preprocessing.

4.2 English to Mandarin Chinese

Below is a table of the most noteworthy models we designed. We then trained our best performing model (RNN GRU) using both training and validation data sets to improve its performance. When tested on Codalab, its scores were the following: Pearson 0.3001, MAE 0.7024, RMSE 0.8887. One may note that the scores were slightly worse on the test data set, especially for the RMSE which went from 0.8483 to 0.8887.

Table 1: Scores on the English-Chinese validation set of our most noteworthy models

Model	Pearson	MAE	RMSE
GRU RNN	**0.3277**	0.7024	**0.8483**
LSTM RNN	0.2426	0.6938	0.8709
Feed-Forward NN	0.3205	0.7053	0.8668
SVM rbf	0.3167	**0.6557**	0.8991
Random Forest	0.2525	0.7130	0.8806

5 Results and challenges

Our results were the best using a RNN with GRU cells. Pearson and MAE scores stayed about the same when evaluating on the dev and test sets whereas the RMSE score went from 0.84 to 0.89 when posted on Codalab.

We encountered multiple challenges during the implementation of our models. At the very beginning of the competition, perhaps naively, we tried to implement state-of-the-art quality estimation sentence translation. We looked through the open source framework OpenKiwi (Kepler et al., 2019), as well as looked at papers proposed for the workshop on Statistical Machine Translation (Bojar et al., 2014; Shah et al., 2015). We realised it would be too time consuming and resized our ambitions with a simple Feed-Forward NN.

Despite this change in difficulty, we encountered more challenges. Beyond the common tensor shapes and dimensions related struggles, the Pearson score was changing dramatically each epoch during training. After looking further into it, one thing we realised is that our network was quickly over fitting to recognise sentence pairs which led to inaccurate scores on the validation set.

We then started to explore RNN architectures. One of the main challenge here was to figure out how to handle two sentences as inputs, when we mostly learnt about RNNs handling one sequence. Our idea was to use two sequences of RNN cells (LSTM or GRU) in "parallel". A fully connected layer would collect the two streams to output the predicted score. This brings another challenge; how to effectively collect the output of two RNN streams? One possible solution would be to take the last step of the two cells and concatenate them side by side to feed to the fully connected layer. This led to poor results. Another solution might be to take the mean or maximum of all time steps for the two sentences and concatenate them. Taking the mean led to the best results for us. One future improvement for the architecture would be to add an attention layer to learn a weighted average.

An issue we had early on is that no matter what architecture we tried, it seemed that after about 20 epochs of training the loss was oscillating and not decreasing. When trying out new architectures we only displayed the loss graph to make sure it was indeed learning. This led us to believe none of our attempts to design an RNN were correct. After displaying the Pearson score at each epoch we realised it was going up, which was good news. The loss in fact decreased very slowly and could be noticeable after about 50 epochs. We did not think to push the training for that long when first attempting to write a learning network.

One of the final challenges we encountered was to write a loss function to improve our scores further. Since the Pearson score is the main scoring metric, we though about using it directly as our loss function. It resulted in networks that very quickly had medium-high Pearson scores, but which did not improve much after the first couple of epochs. After evaluating our networks using other loss functions such as the L1 distance to optimise the MAE scoring metric or even averaging between different loss functions, it turned out the RMSE loss led to our highest Pearson scores.

References

Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Fábio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André F. T. Martins. 2019. OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics–System Demonstrations*, pages 117–122, Florence, Italy. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Terence D Sanger. 1989. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473.

Kashif Shah, Varvara Logacheva, Gustavo Paetzold, Frédéric Blain, Daniel Beck, Fethi Bougares, and Lucia Specia. 2015. Shef-nn: Translation quality estimation with neural networks. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 342–347.

A Appendix

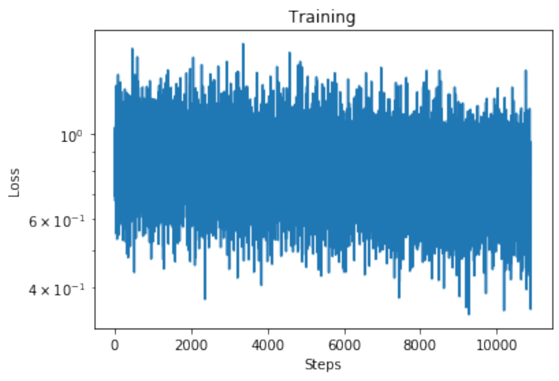

Figure 5: Loss of the RNN GRU, our best model, subtly decreasing