Translation Quality Estimation (English-Chinese)

Yichong Chen
CID / 01790492
yc3919@ic.ac.uk

Yiyang Li
CID / 01753799
yl1319@ic.ac.uk

Abstract

In this coursework, we introduce 3 different models to predict the translation quality giving the English-Chinese translation pairs. We first investigated on a simple feed forward neural network. Data preprocessing was done before training the model with the data. Then we worked on more complicated models such as convolution neural network (CNN), ResNet and recurrent neural network (RNN).

1 Introduction

Translation quality estimation is an important function on machine translation. The computer can estimate the quality of a translation and improve the translation algorithm by maximise the estimated quality. A good translation quality prediction can make a great impact on machine translation algorithm. In this report, we introduce several models to predict the quality score of a English-Chinese translation pair. The training set contains 7000 different translation pairs and also the quality scores of each translation pair were given by human. Moreover, a validation set with 1000 samples will be used to evaluate the model's performance and hyper-parameters tuning. Finally, the trained model will be used on the test set to get the final result on CodaLab.

2 Model 1: Baseline model

In baseline model, data preprocessing is done by applying a tokenizer, removing stop words and doing lemmatization. Each word is represented by a 100-dimension vector and then it takes the average of word vectors so that a sentence is represented by a 100-dimension vector. Support vector regression (SVR) and Random forest were used as a baseline model. As showed in the result, these models only achieve a 0.07 pearson. We think these weak performance result from the unreasonable preprocessing. The preprocessing method we used here removes too many words so that it removes too much information in a sentence. Particularly when we took the average to represent a sentence, the word order is ignored which is important in machine translation. So in the following model, we try to use different word embedding method or make some modification on this approach.

3 Model 2: Fully Connect Neural Network

3.1 Data preprocessing

Here we used another two packages to do sentence embedding, `sentence transformers` [2] is used to embed English sentences and `Google BERT` [1] is used to do Chinese embedding. The word embedding dimension of English and Chinese are 512 and 768 respectively. So an English sentence will be represented by a 512-dimension vector and a Chinese sentence will be convert to a 768-dimension vector. Then two languages are concatenated as a 1792-dimension vector. So the input shape is $(7000, 1792)$.

3.2 Model Design

We used `Keras` to design a fully connected neural network. The structure is showed in figure 1.

3.3 Hyper-parameters tuning

We used grid search to do hyper-parameter tuning and the functions are provided by `scikit-learn`.
1.Optimizer: `Adam` (also tried: SGD, RMSprop, Adagrad)
2.Parameter initialization: `Normal` (also tried: uniform, glorot uniform)
3.Number of epoch: `5` (also tried: 10, 20, 50)

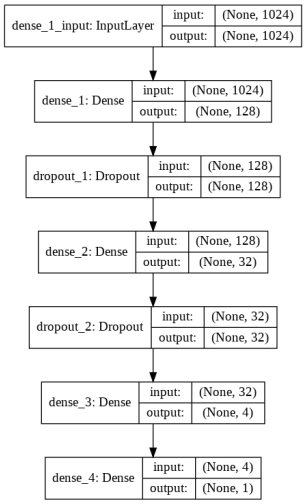

Figure 1: Fully connected neural network

4.Batch size: `64` (also tried: 32,70,100)

3.4 Model performance and analysis

Although this model can achieve a pearson of 0.418 on validation data and 0.426 on test data after 5 epochs, we consider that this model is not sufficiently good since it is easy to overfit the data during training. If we trained the model with more than 10 epochs, it would perform badly on the validation set, even though the training error was still decreasing.
The performance is better than the baseline and we think that this is due to different word embedding methods.

4 Model 3: Convolution Neural Network

In this section, two more complicated neural network models were designed to solve the problem

4.1 CNN model 1

4.1.1 Data preprocessing

Here we tried to use word embedding method provided in baseline and made some modification to make it more sensible. Before put into word embedding, each sentence is padded with <pad> to the same length(50). Word embedding dimension is 100 (<pad> is represented as 100-dimension zero) , so each sentence is represented as a vector with shape(50,100) rather than taking the average of the word vectors. Then we concatenate English and Chinese in one array in a new dimension, like two different channels in image. Finally, the input shape is $(7000, 2, 50, 100)$

4.1.2 Model Design

We use `Keras` to design a Convolutional Nerual Network. The structure is showed in figure 2.

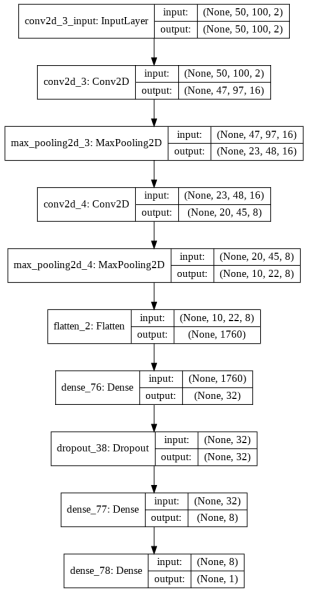

Figure 2: Convolution Neural Network

4.1.3 Model performance and analysis

After 10 epochs training, the model can achieve 0.24 pearson score on validation set but it is difficult to choose hyper-parameters to achieve a higher performance. This method is weaker than fully connected neural network because we think the word embedding method is not powerful in this application. The challenge is to design a word embedding method to represent a sentence and combine two languages reasonably.

4.2 ResNet CNN

Since ResNet convolution neural network can be a powerful model on image classification. We can also apply this model on translation quality estimation problem. By using word embedding, the sentence an be converted to a matrix where each row is the word vector. Therefore, each sentence can be seen as a picture with only one channel.

4.2.1 Model Design

A ResNet CNN is contained with residual block which contains two convolution layers. The input data pass through these two convolution layers and then combine with the input data. The output of a residual block will contain the information from both input data and the output of the convolution layers. In our model, we used two ResNets with

the same architecture to process English cropus and Chinese cropus respectively. We applied three residual blocks in each network and a maxpooling layer was follow after each residual block. After the ResNet, the language outputs will be summed together and pass to a simple feed forward neural network will only two layers.

4.2.2 Performance and Test Result

This model suffer the same problem as before which is overfitting the data. The model can improve the pearson score and the validation loss in the first 5 epochs. Thus, we will only train for 5 epochs with this model and it achieves 0.24 pearson score. After we applied this model on the test set, we got a pearson score of 0.2669. The model does not perform as good as we thought.

5 Model 4: Recurrent Neural Network

In this section, we used recurrent neural network to build a language model to predict the quality of the translation. Due to a fact that human translate a sentence by translate from the first word of the sentence to the end. Therefore, we considered the sentence as a time sequence so that recurrent neural network would be a reasonable idea to solve this problem.

5.1 Preprocess

To apply the recurrent neural network, we needed to preprocess the sentence cropus. In this model, $gensim$ and $glove$ packages to tokenize the sentences. Also, we build a English vocabulary and Chinese vocabulary base on the English training cropus and Chinese training cropus. In the training set, the maximum length of English sentences and Chinese sentences are 36 and 20 respectively. In our model, the recurrent network can only deal with fix length sequences. Thus, we also used "<pad>" as padding to make all the sentence to the maximum length which means all the Chinese sentences have 38 words and English sentences have 20 words.

Now we have a sentence list for each language and each sentence is a word list. Next step we need to transform the words into a vector which is embedding. We have applied two different embedding methods. The first one is add an embedding layer in the neural network. To use this method, we need to build the English and Chinese vocabulary independently. Every word has it's own index number so that we can transform the word list into vectors which the number in the list is the index number of the corresponding word. The second method is applying the pre-train embedding model. The state-of-the-art pre-train embedding model is the google Bidirectional Encoder Representations from Transformers (BERT) [1]. The BERT we use here are the BERT-Base Uncase model and the BERT-Base Chinese model. Both of these model will embed a word into a 768 dimensions vector.

After preprocessing the data, the input shape of the English cropus is $(7000, 20, 768)$ and the input shape of the Chinese cropus is $(7000, 36, 768)$.

5.2 Model Design

The model structure of these embedding is the same. The only different is that we add an extra embedding layer before we use the recurrent neural network (RNN). We have used two different RNN which are long short term memory (LSTM) and gated recurrent units (GRU). For both of these two recurrent neural network, we design the hidden dimension is 1024 and 2 hidden layers. For each language, we used a RNN unit to process the input data which means we have two RNN different units in the network. The sequence length of English and Chinese are different, so we calculate the mean of each RNN output and sum together and apply an linear layer with to get the predictive quality.

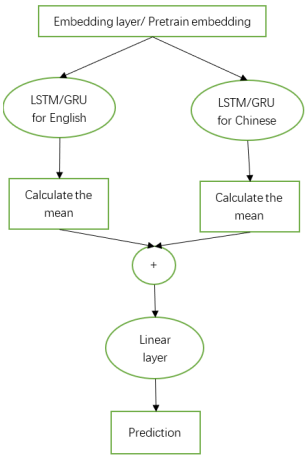

Figure 3: Recurrent neural network

5.3 Hyper-parameters selection

In order to fit the input format for the RNN units, we need to make sure that the number of sample in each batches are the same. So the batch size is 70. For other hyper-parameters, we do the grid search

and find the best combination which the learning rate is set to 0.0002 and use 2 hidden layers with 1024 neurons. To avoid the overfitting, we train the model for 6 epochs.

5.4 Performance and Test Result

After trying both embedding methods on LSTM and GRU network, we find that the pre-train embedding which is BERT has a better performance with the GRU network. Thus we will only analyse and test on this model. The figure 4 shows

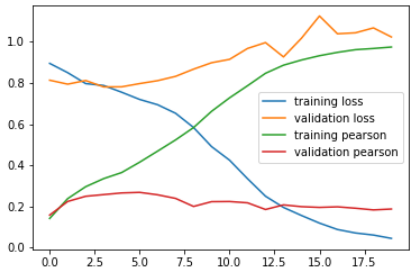

Figure 4: The loss and pearson score of training set and validation set

the loss and pearson score for the training set and validation set in each epochs. During the first 6 epochs, both of the training loss and validation loss decreased and the pearson scores were increase. However, after 6 epochs, the loss and pearson score of the validation set stopped decreasing and started to increase which means the model was overfitted. Therefore, we only train the model with 6 epochs. Finally, this model was used to predict the translation quality of the test set. The model gave a reasonable prediction with 0.33 pearson score and 0.6652 mean absolute error (MAE).

6 Conclusion

Five Models were introduced and all of them work at least better than the baseline model which is SVR and Random Forest. However, they did not work as good as we though. The simplest fully connected neural network gives the best result with 0.426 pearson score and 0.8431 root mean square error (RMSE). The RNN model with GRU units gives the best MAE which is 0.6652. Different embeddings were used in our models, including pre-train model like BERT and glove and also the embedding layer. We found that the a good embedding method can have a great impact on the model performance. For example, both embedding layer and pre-train embedding were used in the RNN model. The model with BERT embedding provide a better prediction with higher pearson score than the model with only the embedding layer which only has 0.1 pearson score. This situation also exists when we optimise our model's performance in hyper-parameters tuning. Although the convolution neural network and ResNet have better performance than the baseline model, they cannot produce a good estimation with high pearson score compare to the fully connected neural network and RNN.

Model Name	Pearson	MAE	RMSE
Baseline Model	0.0795	0.7024	0.9895
Fully Connected Neural Network	0.4260	0.6681	0.8431
Convolution Neural Network	0.2394	0.7287	0.9043
ResNet	0.2669	0.9976	1.1864
GRU RNN	0.3297	0.6652	0.9082

Table 1: Test result on different model

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

Github repository: https://github.com/chanyikchong/Natural-Language-Process.git