Report for NLP Coursework

Team member
Xiaofei Guo / xg1019
Junchen Zhao / jz2119
Wenjing Cai / wc1619
Colab link: https://colab.research.google.com/drive/1FgAKMwFOVC3jW4i4W9p07bOiNFLkihGq

Abstract

This report focuses on the sentence-level quality estimation task. The task is aimed to design a regressor to develop automatic methods to estimate the machine translation quality. We worked on the English to German translation dataset to build our model and evaluate the result. As the prediction is required to perform on the sentence-level, our system managed to score each translation sentence pair according to the human-assessment on the translation quality. For the task, our system architecture consists two main parts: data pre-processing and regressor model. We employed pre-trained word embedding via BERT module [2] to do the word representation and then compute the sentence embedding. As for the regressors, we developed two different models, LSTM [4] and CNN[3], to do the prediction task. The performance of system is measured on three evaluation criteria: Pearson, MAE and RMSE. CNN is chosen as the final regressor of our system according to its Pearson correlation. The final submission result of CNN is 0.162 for Pearson and 0.552 for MAE and 0.933 for RMSE. Finally, our report discussed the different results given by different regressors and encountered challenges.

1 Model design

1.1 Architecture

Our model contains four parts: Pre-processing, word embedding, sentence embedding, and the regressor.

We trained two models with different regressors to compare the results and choose the best one (CNN).

- CNN: Convolutional Neural Networks [3]
- LSTM: Long Short-Term Memory Neural Networks [4]

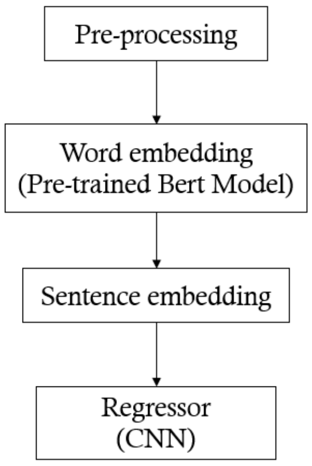

Figure 1: Architecture of the model

1.2 Hyperparameters

The main hyperparameters we adjusted in this coursework are those used in LSTM and CNN.

- Number of hidden states in LSTM

  The number of hidden states is set to 300.

- Epoch number in LSTM

  A large epoch number will cause overfitting, so epoch number of 10, 15, 20,25 were tested. According to the validation performance, the model with 10 epoch was selected.

- Kernel size and number of input channels and output channels in CNN

  Several combinations of these hyperparameters were tested, and the one that get best performance was chosen. The basic idea for choosing channel number is setting them as numbers to the powers of two.

- Epoch number in CNN

  We performed evaluation on validation set for every training epoch, and select the epoch number (9) that reached the best Pearson‘s correlation.

- Loss function in LSTM

  We designed different loss functions for LSTM regressor to compare the model performance. Tradition loss functions for prediction task are usually employed with MSE and RMSE. In our task, considered the primary measure metric is Pearson correlation, we tried the loss function of Pearson for LSTM to do the comparison.

2 Data

Perform the machine translation model on the English to German dataset. The whole dataset is composed by three parts: training set to build models, validation set for model selection and hyperparameter tuning, and test set. The composition of the dataset will be briefly analysed in the following.

For training dataset, there are total three parts: dataset of English text as translation source, dataset of German text as translation destination, and the scores dataset. The scores set describes translation quality for each sentence pair which are used as target ”labels”. The corpus contains 7000 sentences.

Source	José Ortega y Gasset visited Husserl at Freiburg in 1934.
Translation	1934 besuchte José Ortega y Gasset Husserl in Freiburg.
Score	1.1016968715664406

Table 1: Training dataset compostion

2.1 Data Pre-processing

For data pre-processing, we employed BERT module to do the tokenization and extract word embedding[2].

BERT[2] BERT’s key technical innovation is applying the bidirectional training of Transformer to language modelling. BERT takes in vector of tokens as input and outputs predictions dependent on the task objective.

Data preprocessing This part is to pre-process the dataset and make it meet the requirements of BERT model.

- Extract all sentences from dataset.

  Read all 7000 sentences in the training set which are later used for tokenization and word embedding.

- Tokenization

  Using the BertTokenizer to do the word tokenization on the whole dataset. Meanwhile, a $[CLS]$ token is inserted at the beginning of the first sentence and a $[SEP]$ token is inserted at the end of each sentence to build the standard BERT word embedding form.

- Segment embeddings

  These are vectors that indicate if a token belongs to sentence A or sentence B. In our model, since we only consider put the single sentence into the model each time, the segment embeddings are all set to ones vector.

- Extract scores as regressor target

  Get scores from the scores dataset as the target labels which are used for the later regression task. Each translated sentence pair score is computed from direct assessments which is given by human.

2.2 Input Data type

We use pre-trained model ”bert-base-multilingual-cased” via the BERT module as our word embeddings and compute the vector for each word and take the global mean for each sentence.

Word embedding Bert-base-multilingual-cased model is used to build our pre-trained word embedding. It has 12-layer, 768-hidden, 12-heads, 110M parameters. For each sentence, compute the word vector.

Sentence embedding Compute the average word embedding of tokens for each sentence.In practice, sentence embeddings are treated as model input. For each sentence embedding, the dimension is $[1, 768]$. After data pre-processing, the whole source English dataset is transformed into a $[7000, 1, 768]$. Then doing the concatenation of source data and target data to form the actual model training dataset with the dimension of $[7000, 1, 1536]$. Each data entry is actually the concatenation of one source English sentence embedding and one target translation embedding.

3 Regressor

In this translation quality prediction task, we build two regressors to compare the model performance, and choose the model with the higher Pearson correlation.

LSTM[4] LSTMs are very powerful in sequence prediction problems because they're able to store past information.

We built an LSTM model using Keras in this project.

CNN[3] Convolutional Neural networks are widely used in computer vision problems. In the field of Natural Language Processing, it also makes some significant contribution. In computer vision problems, our convolution kernel slides over local areas of the image, and in NLP, the filter slides through a matrix composed of word/sentence vectors.

In this project, we implemented a simple CNN model with three convolutional layers and three fully connected layers using PyTorch.

4 Model performance

Evaluation Criteria To test our system prediction results on translation quality, the evaluation is performed against the true scores using as metrics:

- Pearson‘s correlation (primary)

  Pearson's correlation is a number between -1 and 1 that indicates the extent to which two variables are linearly related. In our task, we use pearson as our primary index for choosing the best model. More close the Pearson vaule to the number 1, the better the prediction system would be.

$$r_{XY}=\frac{\sum_{i=1}^{n}\left(X_i-\bar{X}\right)\left(Y_i-\bar{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i-\bar{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i-\bar{Y}\right)^2}} \quad (1)$$

- Mean Average Error (MAE)

  MAE measures the average magnitude of the errors, the average is over the test samples of the absolute differences between prediction and actual observation

$$\mathrm{MAE}=\frac{1}{n}\sum_{j=1}^{n}\left|y_j-\hat{y}_j\right| \quad (2)$$

- Root Mean Squared Error (RMSE).

  RMSE is the square root of the average of squared differences between prediction and actual observation.

$$\mathrm{RMSE}=\sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(y_j-\hat{y}_j\right)^2} \quad (3)$$

Training Techniques Considering our training dataset maybe not adequate for training a robust model, we first using validation set to choose the best model and tune the hyperparameters. Then we combine the training set and validation set together as the new training dataset to re-train the model with the best set of hyperparameters. The new trained model is our final submission model.

Validation of regressors For both LSTM and CNN, we run the models on specific loss functions and epochs on validation dataset to develop the best set of hyperparameters according to the defined evaluation criteria. The results are shown by Table 2 and Table 3.

Loss function	Epoch	RMSE	Pearson	MAE
MSE	10	0.907	0.166	0.548
	15	0.930	0.133	0.557
	20	0.940	0.153	0.545
	25	0.960	0.114	0.574
Pearson	10	0.881	0.201	0.495
	15	0.912	0.126	0.547
	20	0.881	0.086	0.514
	25	0.875	0.134	0.517

Table 2: LSTM validation performance

Loss function	Epoch	RMSE	Pearson	MAE
MSE	6	0.853	0.193	0.501
	12	0.868	0.203	0.497
	18	0.854	0.184	0.552

Table 3: CNN validation performance

For LSTM, the prediction system with Pearson loss function and run after 10 epochs performs the best. The Pearson correlation value is 0.201.

For MSE loss function, we can see the value of RMSE and MAE show a general upward trend while Pearson decrease as training on bigger epoch number. It indicates that LSTM regressor overfits after training on the 15 epochs. The same situation happens at model with Pearson loss function.

When choosing Pearson as loss function, the pearson indicator shows a better result than the model with MSE loss function. It is because as the model training, it tends to maximize the Pearson value. Meanwhile, the RMSE and MAE indicators values of model trained on Pearson loss are also lower than the model trained on MSE loss.

For CNN, the prediction system with MSE loss function and trained on 12 epochs performs the best.The Pearson correlation value is 0.203.

5 Analysis of results

To get the final prediction results, the performance of the CNN and LSTM regressor was evaluated on the blind test set.

The final hyperparameter set for CNN is:

- epoch = 12
- optimizer: Adam
- layer: three convolution layers and three fully connected layer
- loss function: MSE

The final hyperparameter set for LSTM is:

- epoch = 10
- hidden states = 300
- loss function: Pearson correlation

In order to build more robust model and achieve better system prediction performance, after the hyperparameter tuning, we choose the best set of hyper-parameters and re-train the model on the origin training dataset combined with validation set. The test is also performed on the models trained on the dataset without combined with validation set.

Model	dataset	RMSE	Pearson	MAE
LSTM	val + train	0.945	0.109	0.545
	train	0.938	0.097	0.533
CNN	val + train	0.954	0.139	0.616
	train	0.933	0.162	0.552

Table 4: Test results for LSTM and CNN

It is clear to see that models trained on the combination of origin training set and validation set tend to have better test result.

Accoring to the Pearson correlation(0.162), we choose the CNN trained on the origin training dataset as the final translation quality estimation system. This is to some extent beyond our expectation that the performance of the CNN model trained on the dataset with combined validation set part did not work better than the model without. A possible guess is that the CNN model is not that robust. As of February 28, our ranking is the second in Pearson, 28th in MAE, 8th in RMSE.

From the result, we can see that LSTM performs worse than CNN. [1].

6 Challenges

Computation speed As is mentioned in Architecture (1.1), we tried two regressors and used CNN for the final version. Meanwhile, we also tried SVR regressor to develop the model for the trial. The library we used to implement LSTM (keras) support GPU computation, while the one used for Support Vector Regression (sklearn) does not support that. So the computation speed for SVR regressor was slow. Further more, during the training process, Google Colab sometimes disconnect, leading to the loss of local variable and thus a long time for training.

Dangling validation results To find the most suitable parameters, we trained the models for several times. During the training process, we found that the model can get different validation results using the same hyper-parameters. Thus some parameters gained high Pearson's correlation at validation set, but performed bad in testing. After fixing the random seed and using clear_session() function in Keras backend library, we mitigated this problem.

CNN loss function We have implemented two loss functions for LSTM, which are the built-in mean square error function, and the negative Pearson's correlation function. While for CNN, we only implemented mean square error function, and failed to do the Pearson's one.

References

[1] G. Arora, A. Rahimi, and T. Baldwin. Does an lstm forget more than a cnn? an empirical study of catastrophic forgetting in nlp. In M. Mistica, M. Piccardi, and A. MacKinlay, editors, *ALTA*, pages 77–86. Australasian Language Technology Association, 2019.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. cite arxiv:1810.04805Comment: 13 pages.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[4] A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network, 2018. cite arxiv:1808.03314Comment: 39 pages, 10 figures, 66 references.