Implement

(Data Import)

In this part, we mainly pre-process the data in the train set, and separate them into train set and validation set for cross validation. Firstly, we use function called stopwords(‘english’) to remove some frequently used English words, then we use the ‘re’ package to remove the useless URL at the end of each tweet. Secondly, we create a function called ‘dictionary’ for counting each unique word, and give each word a ID for one-hot vector matrix. Finally, we generate a word-embedding vectors for matrix by using the function in tensorflow which called as ‘embedding\_lookup()’. In the dimension setting, we set the each word’s vector dimension as 50, and the number we count is nearly 7700.

(Training)

After processing the train data, we begin to build the LSTM-RNN model. Firstly, we set the batch-size, the units of LSTM, the number of classification and the iteration times. We use the function tf.nn.rnn\_cell.BasicLSTMCell for set the units of LSTM we wanted, and this is the hyper-parameter we can use to optimize our model. Then, we set some drop-out parameters for avoiding over-fitting, and some normal parameters like weights, bias and so on. After that, we use a standard cross-entropy loss function. For the optimizer, we choose ‘Adam’ and the default learning rate.

Shortly, the whole training process is, for each twitter, we use one-hot vector multiply with word-embedding matrix, then send it into LSTM training model and get result. With cross validation and the loss function of cross entropy, we train the data sets 150,000 times and get the final LSTM model. Then we use test data to get the prediction of tweets.

(training result)

we can see from the training curve above, we found that the training results of this model is still good. The value of the loss is declining steadily, and the accuracy rate is constantly nearing 100%. However, when analyzing the training curve, we should note that our model may have been over-fitting the training set. Over-fitting is a very common problem in machine learning, which means that the model fits well over the training set, but the generalization ability on the test set will be much worse. That is, if you take a model that has a loss value of 0 above the training set, this result is not necessarily the best result. When we train LSTM, early termination is a common way to prevent over-fitting. The basic idea is that we train the model above the training set, and colleagues constantly measure its performance over the test set. Once the test error has ceased to fall, or the error has started to increase, then we need to stop training. Because of this sign, the performance of our network has started to degenerate.

(test result)

After getting the training LSTM model, we use the data from the test set and get the test result. We upload the result to the Kaggle website, and we can see that the log-loss of model is much lower than the loss of sample-prediction, which is logistic regression model. It shows that the LSTM is a powerful and useful tool for classification tweets.