COMP4332 Group 12 Report

Siu Chun Hin, Wang Liang-huai, Yeung Wai Kit, Fu Yin Ki Carrie

Trial 1: XGBoost

Pre-processing

The data set was first tokenized to remove stopwords, punctuations and stemmed. Then the feature dictionary was mapped.

Define model

Pre-processed data was fed to train RNN, LSTM, BiLSTM and GRU models. LSTM models have the best accuracy result while RNN is the worst. This may be due to lack of indication for beginning and end of sentence. The trained models showed they have room for improvement, so a trained embedding matrix was then introduced using a fasttext library and the embedding layer together with the pre-processed data were fed to LSTM and BiLSTM model to train the models. Results showed that the accuracy remained similar, but there was improvement in stability.

Why don't we include this as the final model?

XGBoost was also used as an optimized distributed gradient boosting library to perform sentiment analysis. The data is first tokenized by TF-IDF and then fed to XGBoost with optimal parameters previously tuned by Optuna. Eventually a result of 56% accuracy was obtained. This relatively suboptimal result may be due to the fact that XGBoost performs more efficiently on binary classification but subpar on multiclass classification.

Trial 2: Bert (Without GloVe embedding layer)

Pre-processing

We picked the pretrained bert-base-cased model which trained on cased English text as we need to classify output into 5 intensity of sentiment. In this case, the word "BAD" might imply a different meaning from "bad". Thus, we treat the analysis as case-sensitive. For the text part, we tokenize using the pre-trained model and standardize all text into a maximum length of 500. We standardize them by using [CLS] to mark the start of the sequence, [SEP] to separate sequence/marking the end of the sequence and [PAD] to pad all text sentences to the same length. The tokenizer will give us two arrays. One (input) contains the ID representation of text while the other (attention mask) contains 0s and 1s. The attention mask is to avoid bert making connections with any of the padding tokens. We will then merge the two tensors into a dictionary thus later on the model can intake both arrays as input. For the prediction (stars) part, we convert them into 5 one-hot columns.

Define model & tuning

We set the batch size as 5 thus we have 5 samples in every tensor. First, we created two input layers using input_ID and attention mask. Next, we create the embedding layer and access the pooled activations. Then, we create a 512-unit dense layer followed by a dropout to prevent overfitting. And at last, the output layer of unit 5. We used a softmax activation function so as to calculate the probability across all 5 output labels. We tried different layer units [1024, 512, 256] and found the result of the validation set is quite similar thus we go for the intermediate one. Then, we tried to use optimizers [Adam, RMSProp], to loop through different learning rates [5e-5 to 1e-4] and decay [1e-4 to 1e-2] and at last discovered the optimal value parameter of using optimizer: Adam, Learning rate: 1e-5, decay: 1e-3.

Why don't we include this as the final model?

Bert model with GloVe appears to be more stable than bert model alone as it is trained using a pre-trained corpus.

<u>Trial 3:</u> CNN+GridSearchCV

<u>Pre-processing</u>

We extract the features from the “review texts” like returning the lower characters from the text (used for CNN model), tokenize the text, reducing the words in the text to its word stem and return a contiguous sequence of 2 and 3 items from the “review text”. Then build the feature list and build the mapping from features to indices. Lastly, convert the features and labels(stars) to matrix.

<u>Define model</u>

We use 1-layer perceptron (1 dense layer with softmax activation) classification model with grid search hyperparameters using keras model - “sequential”. For the grid search hyperparameters, we use “batch size” and “epochs” for searching the best combination for the optimized results. We train the model with different configurations (4 x 5). Grid search uses the model creating function to build a model for each hyperparameter. Then choose the best “optimizer” for the model using the best parameters of batch size and epochs.

Then extract these optimized parameters (batch size, epochs and optimizer) to the CNN model. CNN passes the feature matrix through the convolution layer, ReLu layer, pooling layer and then apply the activation function to classify the probabilistic values between 0 and 1. Also, adding the dropout to avoid some instability and over-training problems.

<u>Why don't we include this as the final model?</u>

The result is worse than the Bert+Glove model because there are no pre-trained texts for word embedding and also the validation loss is quite large compared to the Bert+Glove model. The accuracy can only reach about 58%.

<u>Trial 4: GloVe with LSTM and GRU</u>

<u>Pre-processing:</u>

We extract the features from the “review texts” like returning the lower characters from the text , tokenize the text, and transform it to sequence with a fixed length of 200.

<u>Define model:</u>

This model is referenced from online resources. The first layer is an embedding layer with pretrained GloVe features. The pretrained embedding we used is from twitter as we consider twitter review is similar to the review we are processing. The model consists of one spatial dropout 1D layer, one bidirectional LSTM layer, one bidirectional GRU layer followed by two separated pooling layers. The two pooling layers are concatenated for the final output layer. The parameters in this model are tuned with Optuna.

<u>Why don't we include this as the final model?</u>

The result of this model can reach an accuracy of around 61%. We later tried to combine this model with BERT and the result is better and more stable.

<u>Trial 5: BERT + GloVe</u>

<u>Pre-processing:</u>

The pre-processing is the same as Trial 2 and Trial 4. In addition we combined the two pre-processed dataset into one for input.

<u>Define model:</u>

The architecture of the this model is referenced from online resources. There are two input layers, one is BERT and the other is GloVe embedding. For the GloVe embedding layer, the architecture is the same as Trial 4 except there is only one pooling layer. The pooling layer is concatenated with the BERT layer followed by a dropout and final output layer. The parameters in this model are tuned with Optuna.

The result of this model can reach an accuracy of around 65%. It shows that the combination of GloVe embeddings and BERT gives the most accurate prediction. Therefore, this is our final model.