CS583 – Research Project

Rishabh Mehta (rmehta35) & Pramodh Acharya (pachar7)

This project aims to analyze the sentiments of tweets about the presidential candidates and classify them as positive, neutral, or negative sentiments. The data is from Twitter during a presidential debate between Obama and Romney. The resultant classification is a probable measure of the election outcome.

We created a sentiment analyzer to classify tweets in text format as positive, negative, or neutral. We followed the following set of steps to achieve the same.

The Dataset

The training data is a set of labeled twitter data with the tweet text, date, and timestamp of the tweet.

The test data is a set of unlabeled data with each row containing a tweet id and the text.

Text Preprocessing

Several steps of standard processing for Twitter data were performed on each row of raw data. Following are the sequential steps of preprocessing used.

- **Cleaning Raw Tweets**
  - *Lowering Case:* Lowering the case of all the words helps to reduce the dimensions by decreasing the size of the vocabulary.
  - *Removal of Html tags:* Removed anything enclosing a '<' and '>' inclusive. This removes any Html DOM elements as these are not useful in training the model.
  - *Removal of Hyperlinks:* We remove URLs by searching for any part of the sentence starting with an 'http' and removing the complete URL string.
  - *Removal of mentions:* We remove mentions from tweets as they are not useful. Any word or part of the sentence starting with '@' is removed completely.
  - *Removal of Special Characters:* All remaining special characters like '#' or punctuations like ',', '.' are removed.
  - *Removal of Numbers:* Numbers in most cases do not provide any meaningful opinion on the sentiment and hence we removed it.
  - *Removal of Stop words:* Stop words were removed with help from nltk.corpus English library's 'stop words' as they don't provide valuable information towards the analysis or sometimes skew the model.

- **Tokenization**
  This is the process of splitting the processed raw tweets (sentences) into smaller chunks such as a list of words. We initially tried the spacy module to tokenize and lemmatize the sentences, but splitting it manually (without any library) and using wordnet to lemmatize provided better results.

  A sub-step while tokenizing is splitting any joined words, which is mostly the case in hashtags. This is done by using a utility function we wrote to split any possible word to the least dimension possible.

- **Lemmatization**
  This is the process of fetching the lemma's or the simplest form or root form of the words as present in a dictionary. We lemmatize all the verbs and nouns/plurals. This reduces the dimensionality of the dataset by a large value.

Models and Techniques Used

- **Word Embeddings**
  Word Embeddings were generated using 200 dimensional Glove vectors that were pretrained on 2 billion tweets.Word Embedding matrix were generated using the Glove vectors and training data with dimensions (len(train_data) +1,200) additional token is added to consider out of vocabulary words.This matrix will later be used as embedding layer weights in a Sequential Model.
- **Sequential Models**
  Sequential models like LSTM,GRU,RNN and BiDirectional LSTM were used for model training
  - *General Architeture of the model:* Input layer consists of a data vector , which padded to ensure length of each vector is 50,following this is a embedding layer whose weights have been fixed using the embedding matrix which produces a output of size (50,200) dimensional vector which is then forwarded to a Sequential layer which contains either 100 or 200 units and can be stacked and was L2 regularized .Output of this Sequential layer is sent to a Dropout layer to prevent overfitting and then final layer is Dense layer with 3 outputs and sigmoid activation.
  - *Model Weights:*The models had checkpoint enabled which only stores the best weights(monitored on validation loss) in all the epochs and at end of epochs these weights were restored as model weights.
  - *Model Evaluation:*Various Combinations of different Sequential models were run using different units and stacking the sequential layers. Validation Loss was used a metric to evaluate model training.If the model had early stopping enabled training stops at the point where validation loss no longer decreases this prevented the model to be more generalizable and not overfit on training data.Along with validation loss classification report and confusion matrix were generated over validation data to further confirm the training performance.
  - *Model Selection :* After training over multiple models with different permutations of parameters we found the following model to achieve best performance which is GRU with 100 units. Figure 1 shows the architecture of the best model.

```
**GRU early_stopping=True,stacked_model=False,stack_length=1
**** Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding (Embedding)        (None, 50, 200)           2181800   
_________________________________________________________________
gru (GRU)                    (None, 100)               90600     
_________________________________________________________________
dropout (Dropout)            (None, 100)               0         
_________________________________________________________________
dense (Dense)                (None, 3)                 303       
=================================================================
Total params: 2,272,703
Trainable params: 90,903
Non-trainable params: 2,181,800
_________________________________________________________________
```

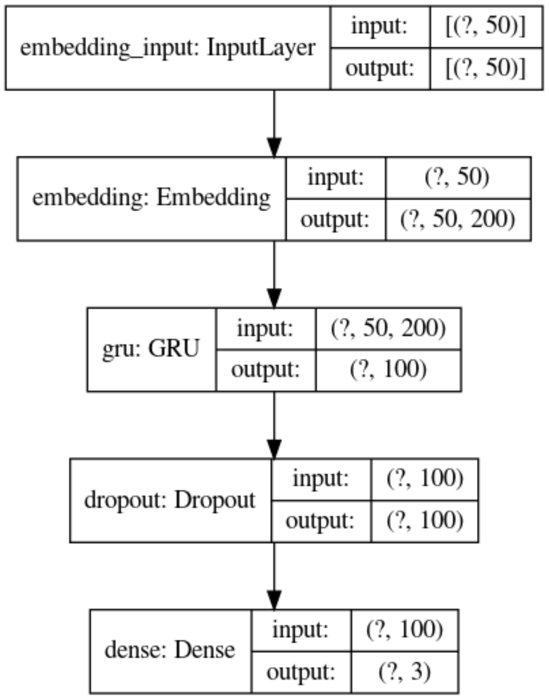


Figure 1 Model Architecture

Experimental Results

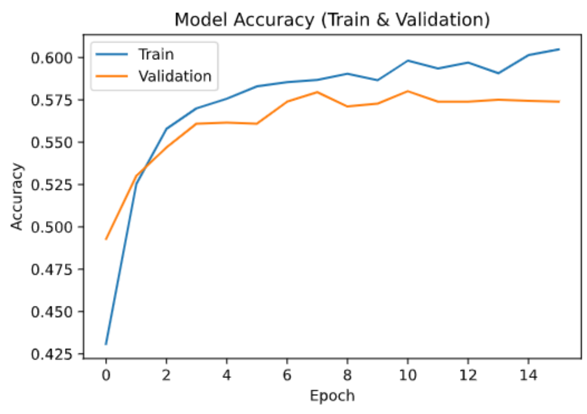


Figure 2 Model Accuray(Train and Vaidation)

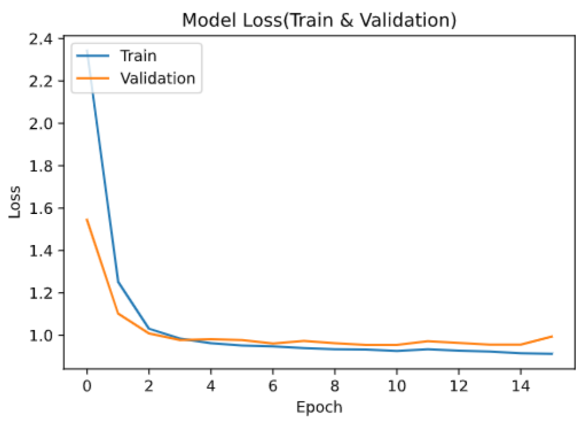


Figure 3 Model Loss(Train and Validation)

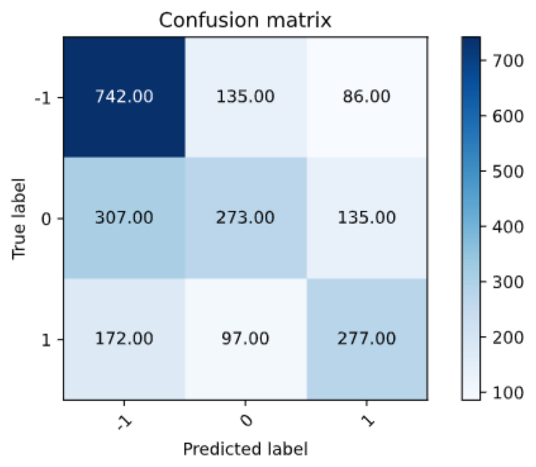
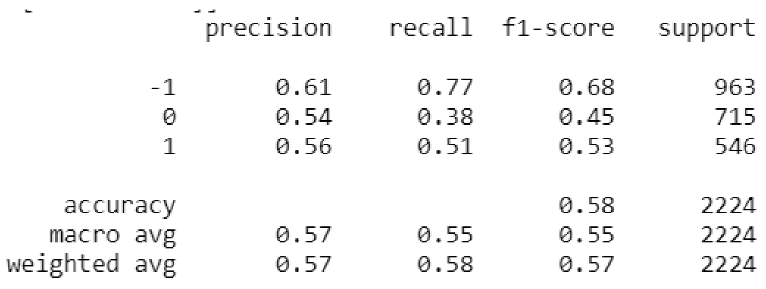


Figure 4 Confusion Matrix



```
              precision    recall  f1-score   support

          -1       0.61      0.77      0.68       963
           0       0.54      0.38      0.45       715
           1       0.56      0.51      0.53       546

    accuracy                           0.58      2224
   macro avg       0.57      0.55      0.55      2224
weighted avg       0.57      0.58      0.57      2224
```

Figure 5 Classification Report

- **Model Training**
  Training results for the above selected model are as follows the above plots(Figure 2 and 3) show training and validation accuracy and loss plots ,with training-loss: 0.9254 - training -categorical_accuracy: 0.5982 val_loss: 0.9540 - val_categorical_accuracy: 0.5801 for the best epoch. We also generate confusion matrix (Figure 3)and classification report(Figure 4) which shows the accuracy , precision ,recall and f1 score for each predicted class.
- **Prediction**
  The best model GRU was then re trained on the whole dataset and the model was saved , which was then later used to predict the sentiment for the test data set,which was preprocessed similar to training dataset and converted to vectors.The final prediction results are stored in Obama.txt and Romney.txt

Conclusion

- Early stopping in sequential models was very important ,without early stopping many models would overfit and validation loss would start increasing after a certain number of epochs.
- Best learning rate found for training these models on this data was 0.01

References

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras