# **Midterm Project: Sentiment Analysis of Twitter data and News Headlines**

Team: Anuja Kapre and Raksha Kaverappa

Github repository: <https://github.com/raksha592/Sentiment-Analysis-Midterm-Project>

For the data provided, the problem was divided into two subtasks:

1. Sentiment Analysis of twitter and stocktwits data
2. Sentiment Analysis of News Headlines

For each of the subtasks, the following operations were carried out to obtain adequate accuracy and cosine similarity. The process was as follows:

1. Data Pre-processing
2. Split data into train and test
3. Tokenization of the words
4. Building the model
5. Compare the predicted sentiment score with the actual sentiment score
6. Obtain the cosine similarity
7. Predict sentiment score for test data

**Data Pre-processing:**

***Sub-task 1:*** For the twitter and stocktwits data, the data was cleaned considering various factors to improve the prediction accuracy. First, the keys containing the messages (‘spans’) was converted from lists to strings to help access the values of strings more easily. Next, the data in the dictionary was converted to lowercase. The punctuations and the unwanted characters were removed to clean the messages. To improve the prediction of the sentiment score, stopwords were removed from all the messages using NLTK library.

***Sub-task 2:*** For the new and headlines data, the data was cleaned to remove the company name form the messages and replace it with a blank. Next, the keys containing messages were converted to lowercase in the dictionary. Punctuations and non-ascii characters were removed to clean the strings and the stopwords were removed using NLTK library. These factors and preprocessing steps helped improve the prediction of the sentiment score in the analysis.

**Split data into train and test:**

For the first step of the evaluation, we combined the Training data and the trial data into one complete training dataset.

We evaluated the model and tested the same model to compare the sentiment score to the predicted sentiment scores.

X: Messages/Spans

Y: Sentiment score

For the second part, we trained the model with the training dataset and tested it with the test dataset provided. The same data pre-processing was carried out for the test dataset.

**Tokenization of the words:**

We used the keras inbuilt library to tokenize the strings into words. We used the pad\_sequences library from keras to convert the tokenized words into a numpy array of having the number of rows as the maximum length of the words in a string.

Sub-task 1: While padding these sequences, since dataset was larger, the maxLen attribute was set to the length of the longest sentence of the dataset (25). Also in tokenization, the num\_words attribute was set to maxLen \* 20 (500).

Sub-task 2: While padding sequences, since dataset was smaller, using the maxLen attribute as the length of the longest sentence of the dataset was overfitting the model.

The problem was solved by using average no of words per sentence (7) as the maxLen.While tokenizing, the num\_words attribute was set to maxLen \* 50 (350).

The same operation was performed for the test dataset.

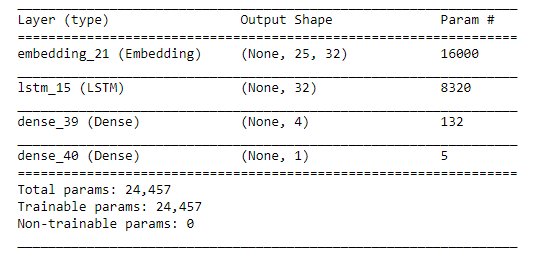
Building the model:

We implemented the model for both the datasets using two different models. The first model was using RNN and the second model was using CNN. We tested the model by changing several hyperparameters and observed the following:

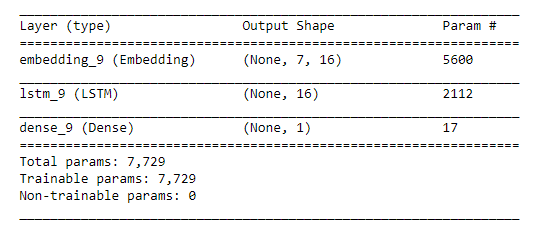
* The model predicted very low accuracy of 3.75% with both RNN and CNN when the activation function was relu or softmax
* The model had good accuracy for activation function sigmoid, however, it was stuck and was unable to give proper prediction sentiment score
* The activation function was changed to tanh and we obtained a prediction accuracy above 60% using RNN and CNN for both the tasks
* We changed the dropout rate from 0.01 to 0.3 to get an improved accuracy
* The model for CNN was predicting maximum positive value (0.99) for all the messages having positive sentiment score.

Based on the observations above, the RNN performed best giving the best accuracy and an accurate prediction of sentiment score. Our final model summary is as shown below:

***Sub-task 1***:



***Sub-task 2***:

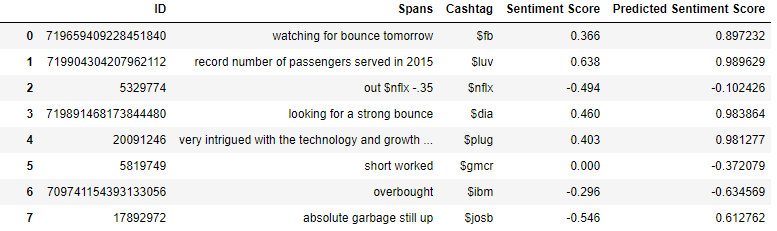


**Compare the predicted sentiment score with the actual sentiment score:**

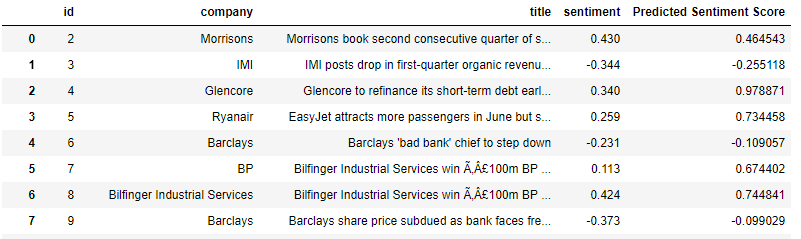
The predicted sentiment score and the given sentiment scores were compared after testing the model on the training dataset.

Following are the results of predicting sentiment score for training dataset for task1 and task2

***Sub-task 1***:



***Sub-task 2***:



**Evaluate model based on the cosine similarity:**

The cosine similarity is computed using the cosine\_similarity function from scikit learn. The cosine weight is the ratio of normalized vectors of Gold Standard score and Predicted score. The final cosine score observed for both the tasks were:

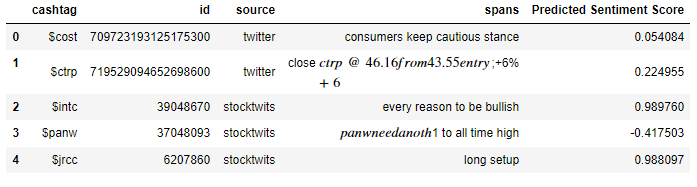
***Sub-task 1***: 0.6917

***Sub-task 2:*** 0.6932

**Predict Sentiment score for test data:**

The predicted sentiment scores for the test dataset was also obtained. The results are as shown below:

***Sub-task 1***:



***Sub-task 2:***

