**Name:** Didier Fernando Salazar Estrada - 利 葉

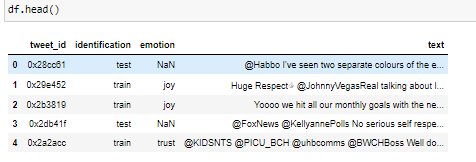
**Student ID:** 111065427

**Course Name:** Data Mining

**Report – Developing my model for the competition**

**Pre-processing and Feature engineering steps:**

The following data was given:

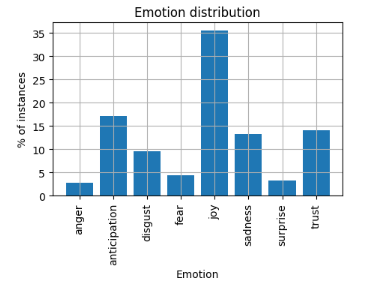


There was a tweet\_id, a form of identification to know what tweets to use to train the model and to test it, the emotion labels and the text of the tweets.

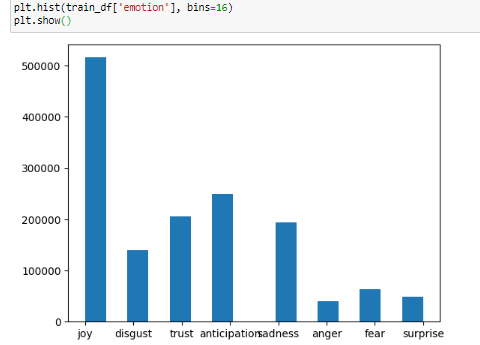
It was divided into train and test data:



Here there are the emotions distribution throughout the whole training set:



This is the number of tweets distributed between each classified emotion:



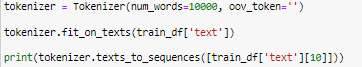
Sampling the data set:



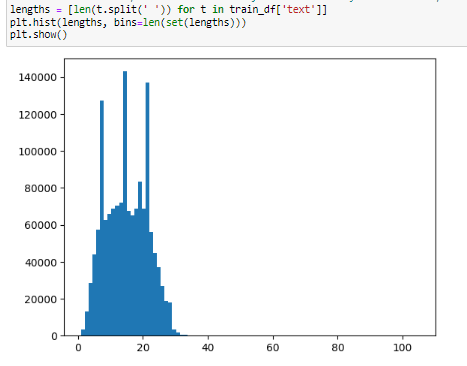
To preprocess the text features of the tweets the built-in tokenizer of tensorflow was used:



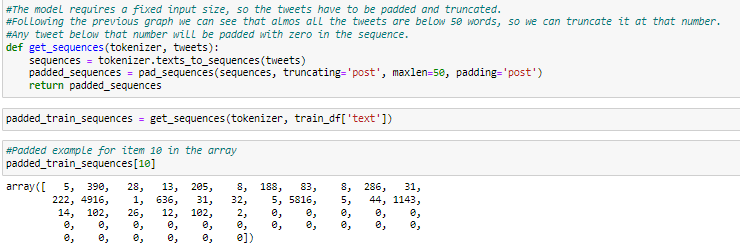
The 10,000 most frequent words were used:



Here is the graph to see what’s the most common number of words in each tweet for the whole training data of the tweets. It can be appreciated that the numbers most common are approximately between 0 – 30.



Because not all the tweets have the same number of words in each corpus, they have to be padded and truncated to a fixed number.

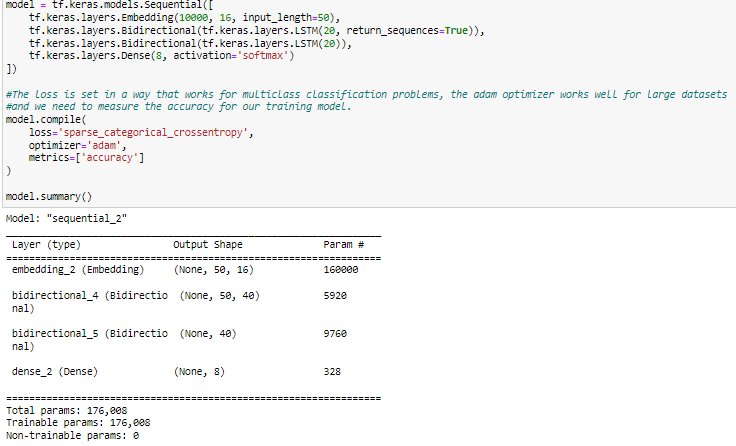


**RNN model:**

To build the predictive model the RNN (Recurrent Neural Network) was chose, they work well with data that it’s in the form of a sequence, in this case the corpus of a tweet to predict the emotions for each of them.

Although one of the disadvantages of it is that suffers from exploding or gradient vanishing if the data is too big, so the training becomes difficult because it doesn’t learn well in that condition. To tackle this problem LSTM (Long short-term memory) was used to support it, the LSTM is bidirectional, this means that run from the past to the future and vice versa, during training, this helps preserving the results from both past and future and patches this disadvantage in the RNN algorithm.

The dimensions are 10,000 for the most frequent used words in the data, 16 for the output for the next layer, input\_length corresponding to the padded and truncated length of each tweet (it was tested with different values). The LSTM layers they both have 20 cells which have their own input, output and memory. The final layer has 8 dimensions which corresponds to the eight emotions that we need to classify, it uses softmax so it will return a probability distribution between each emotion in a tweet.

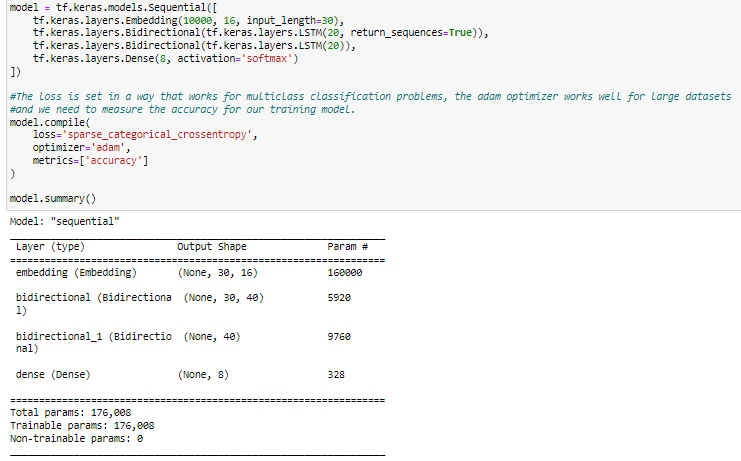


**Model testing:**

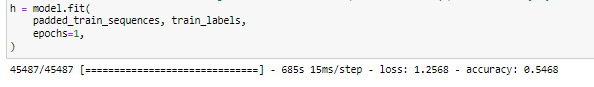
Because the most common length in words for each tweet was between 0-30 it was first tested truncating and padding them to 30.



The model was built with 30 for input-length.



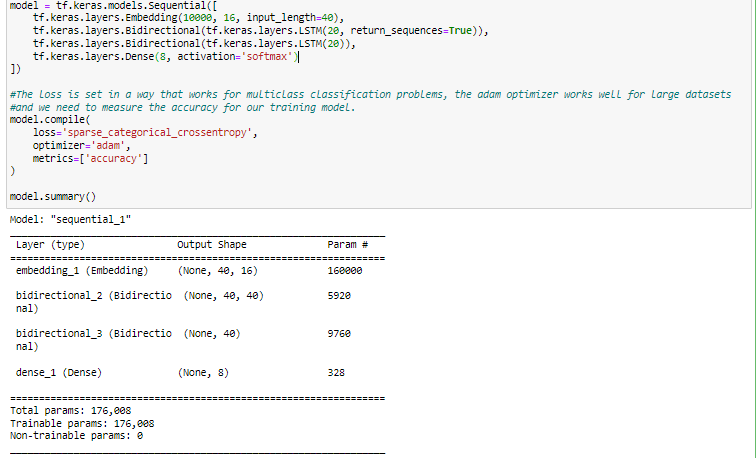
And the result for the training test in the first epoch was a 54.68% in accuracy.



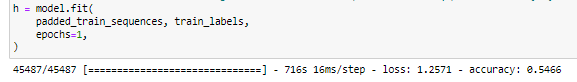
Testing with other input-lengths to see if it changes anything for the better, I tried also with 40 words to truncate and pad the tweets.



The model was built with 40 in input-length.



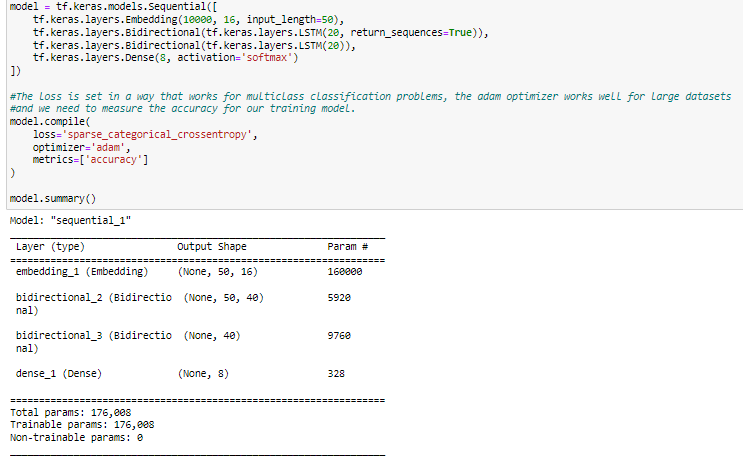
The training test didn’t change much, the accuracy was 54.66%.



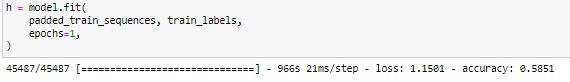
Then I tried truncating and padding to 50 words.



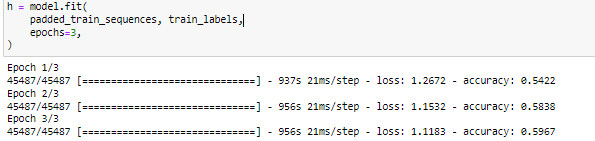
The input-length was changed in the model to 50.



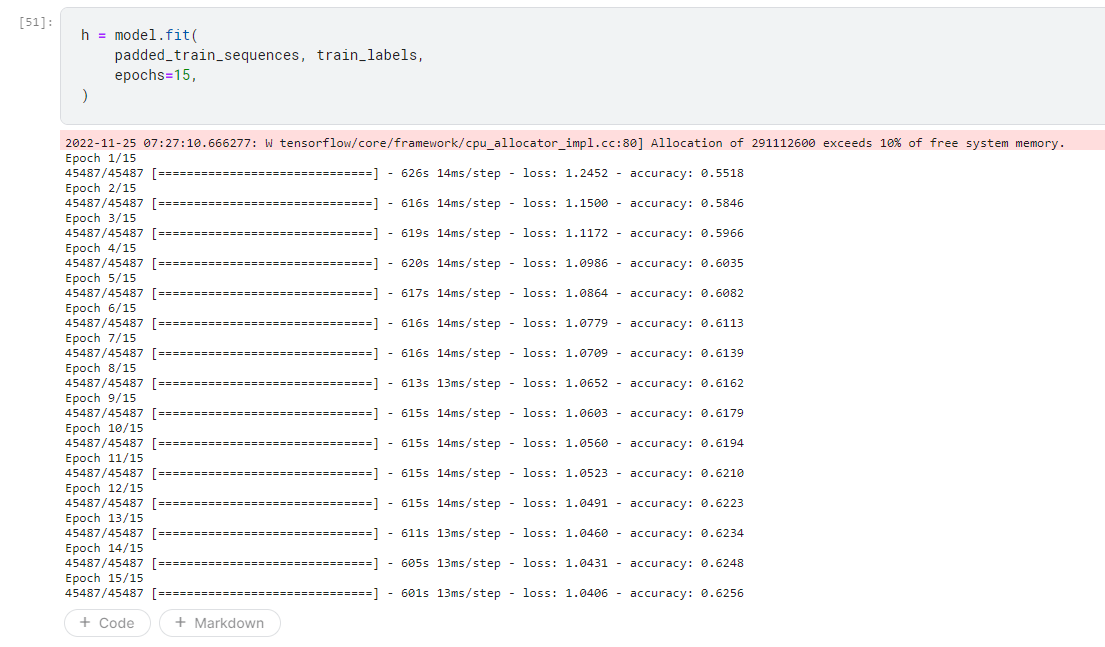
And here I saw an improvement in the accuracy for the first epoch, **58.51%.**



So, using this as a base I tried with 3 epochs and I obtained a **59.67%** in accuracy.



Then I tried using 15 epochs and I obtained **62.56%** in accuracy.



I calculated the predictions for the testing set with the model using 50 words truncated and padded with 1 epoch, 3 epochs and 15 epochs.



Submitting the predicted results with the testing set to the Kaggle competition I saw that the predictions with 1 epoch had **46.57%** in accuracy, with 3 epochs **47.54%** and with 15 epochs **47.77%.**



**My final ranking was in the 32th place for the competition (Friday 25):**

