Twitter Sentiment Analysis

Objective :

Hate speech is unfortunately a common occurrence on the Internet. Often social media sites like Facebook and Twitter face the problem of identifying and censoring problematic posts while weighing the right to freedom of speech. The importance of detecting and moderating hate speech is evident from the strong connection between hate speech and actual hate crimes. Sites like Twitter and Facebook have been seeking to actively combat hate speech. Despite these reasons, NLP research on hate speech has been very limited, primarily due to the lack of a general definition of hate speech, an analysis of its demographic influences, and an investigation of the most effective features.

Here, I have used different deep neural network models to effectively classify offensive comments.

Data :

Train.csv - Labelled dataset of 31,962 tweets. Csv file with each line storing:

1. Tweet id
2. Tweet sentence
3. label '1' denotes the tweet is offensive and label '0' denotes the tweet is not offensive

Test.csv – Each row of test data contains:

1. Tweet id
2. Tweet text with each tweet in a new line.

Metric :

F1 Score is being used to compare the models due to class imbalance.

True Positives (TP) - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes.

True Negatives (TN) - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no.

False Positives (FP) – When actual class is no and predicted class is yes.

False Negatives (FN) – When actual class is yes but predicted class in no.

Precision = TP/TP+FP

Recall = TP/TP+FN

F1 Score = 2(Recall Precision) / (Recall + Precision) F1 is usually more useful than accuracy, especially if for an uneven class distribution.

Text classification using word embeddings and deep learning :

Word embeddings are layers used in neural networks which projects words into higher dimensional vectors. Unlike other text vectorization techniques, word embedding layers trains as like other neural network layers. The words with similar meaning are clustered together at the end of training. Embedding layers understands the meaning of the words from the labels.

MODEL WITH EMBEDDING LAYER AND NORMAL DENSE LAYERS :

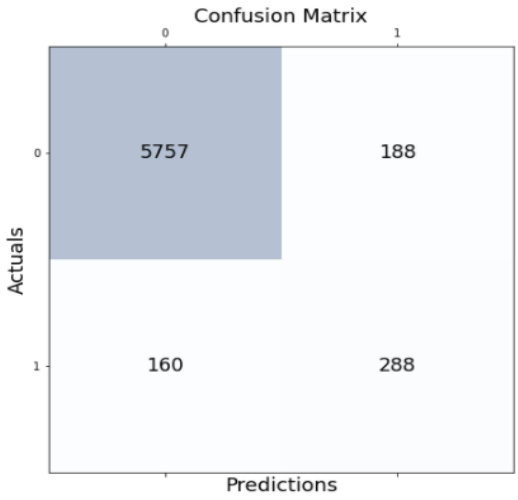
Summary of Sequential model is attached below:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 50, 18)            540000
_________________________________________________________________
flatten (Flatten)            (None, 900)               0
_________________________________________________________________
dense (Dense)                (None, 64)                57664
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 16)                528
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 136
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 36
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 5
=================================================================
Total params: 600,449
Trainable params: 600,449
Non-trainable params: 0
```

Training accuracy: ~96%

Testing Accuracy: ~ 95%. Confusion Matrix plotted based on Test dataset's actual Label Vs predictions is attached below for reference.

F1 Score: 62.33%



We can observe that **False Negative is higher in number**. In order to reduce FN building a bit more complex model.

Model with Embedding, 1-dimensional Convolutions (Conv1D) and Dense layers :

As per the objective, it is important to make sure that the model is able to identify all the offensive comments in Twitter space. So, the **focus is to decrease the False Negatives** rather than accuracy.

For this, I have **added a Convolutional layer** to increase the interpretability of the model to identify all the offensive comments. Summary of Convolutional model built is attached below:

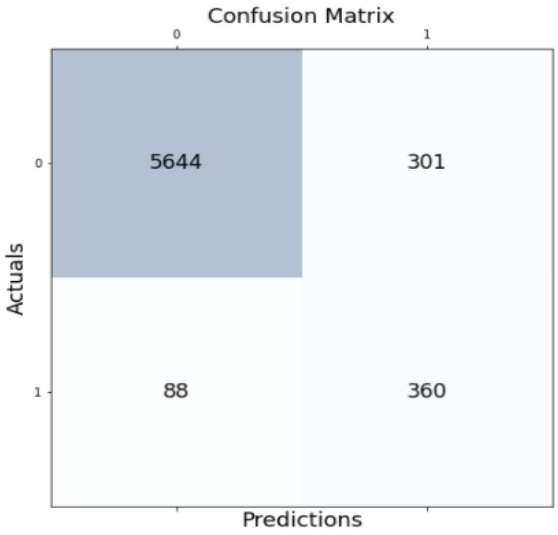
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 50, 18)            540000
_________________________________________________________________
conv1d (Conv1D)              (None, 47, 16)            1168
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 23, 16)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 20, 32)            2080
_________________________________________________________________
flatten_1 (Flatten)          (None, 640)               0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                41024
_________________________________________________________________
dense_7 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_8 (Dense)              (None, 16)                528
_________________________________________________________________
dense_9 (Dense)              (None, 8)                 136
_________________________________________________________________
dense_10 (Dense)             (None, 4)                 36
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 5
=================================================================
Total params: 587,057
Trainable params: 587,057
```

Training Accuracy: ~96%

Testing Accuracy: ~94%

F1 Score: 64.92%

Confusion Matrix plotted based on Test dataset's actual Label Vs Conv model predictions is attached below for reference.



We can observe that **FN has reduced by half the number compared to Sequential model**. We can say that, CNN is effectively identifying offensive comments as per our objective.

LSTM – Long Short-Term Memory

```
Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_30 (Embedding)     (None, 37, 64)            320000
_________________________________________________________________
lstm_30 (LSTM)               (None, 128)               98816
_________________________________________________________________
dropout_36 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_30 (Dense)             (None, 1)                 129
=================================================================
Total params: 418,945
Trainable params: 418,945
Non-trainable params: 0
_________________________________________________________________
```

Training Accuracy : 95.42%

Testing Accuracy: 95.41%

F1 Score: 57.74%

Confusion Matrix based on Test dataset's actual Label Vs LSTM's prediction is attached below for reference.

```
array([[5884,  101],
       [ 192,  216]])
```

We can observe that False positives is higher than other 2 models and False negative is lower.

Conclusion :

Among Sequential, CNN and LSTM models **CNN performs better** as we saw that False Negative Rate is very low which means model is **able to classify maximum number of offensive comments accurately** that will help app to delete such comments more effectively.

Next Steps:

For increasing effectiveness in classifying the tweets, I have planned to combine architectures of CNN and RNN networks.