Real or Not? NLP with Disaster Tweets

San Jose State University

Electrical Engineering Department

(Course: Neural Network)

Date: 14th December 2020

Dhairya Vipul Shah (014602978)

Project Partner

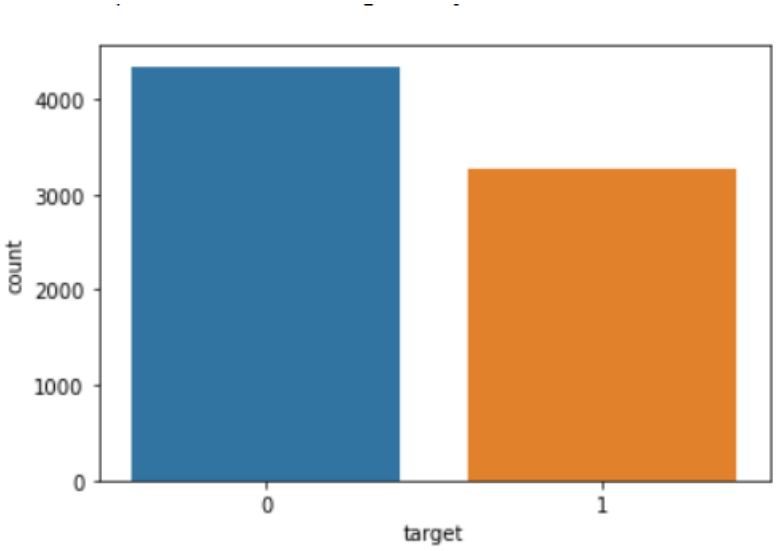
Anuja Sapkal

Table Of content

DATASET DESCRIPTION:

This dataset is part of the kaggle challenge, here one has to predict whether a given tweet is real or not regarding the disaster. To know much about it lets look at the dataset in detail. First of all we need to know why this is required and answer to this is simple that if a particular tweet is fake about any disaster that would create panic amongst the people.

Dataset is divided into two part train dataset and test dataset. Both the dataset is in CSV file. Train dataset has five column in it i.e. ID, keyword, location, text and target. Some keywords are army, ablaze, etc. Location by the words means for which location is the tweet for. Text column actually contains tweets in it. Target gives the information about whether the tweet is real disaster or not. “0” in target column means it is about real disaster while “1” means it is about fake tweet.



Talking more about text column, one can visualize It in two ways like the length of tweet in terms of number of character and in terms of number of words in it.

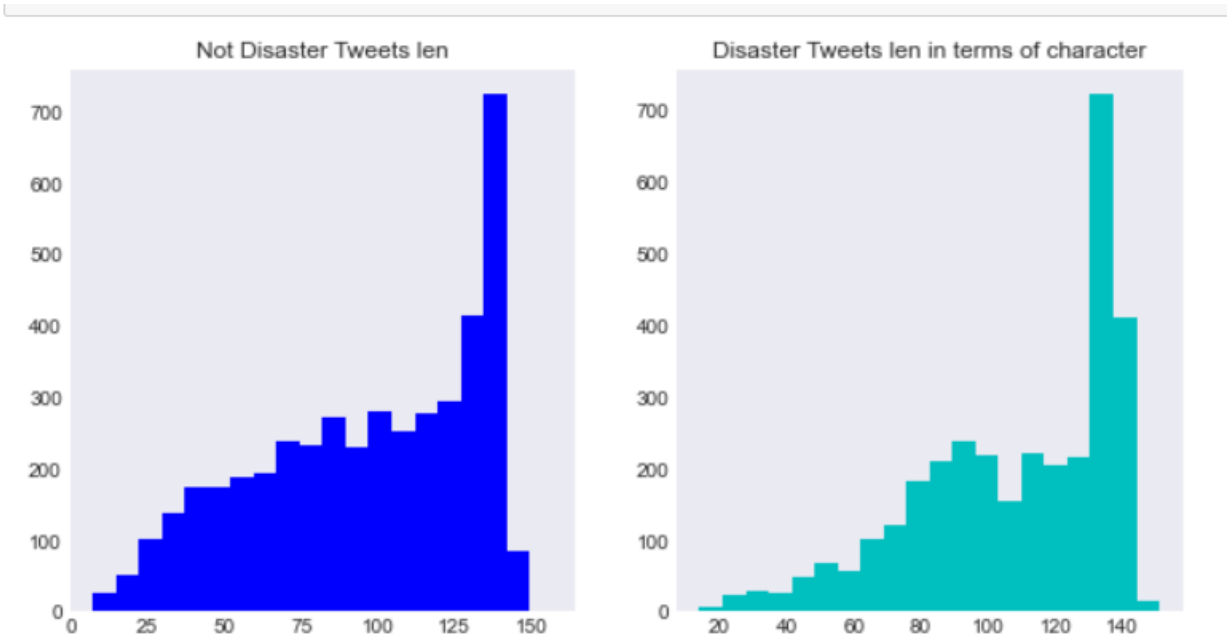


fig-1 Tweet in terms of character length(train dataset)

From the above figure we can clearly shows that the tweet are generally on the larger number of size in terms of number of character be it train dataset and test dataset.

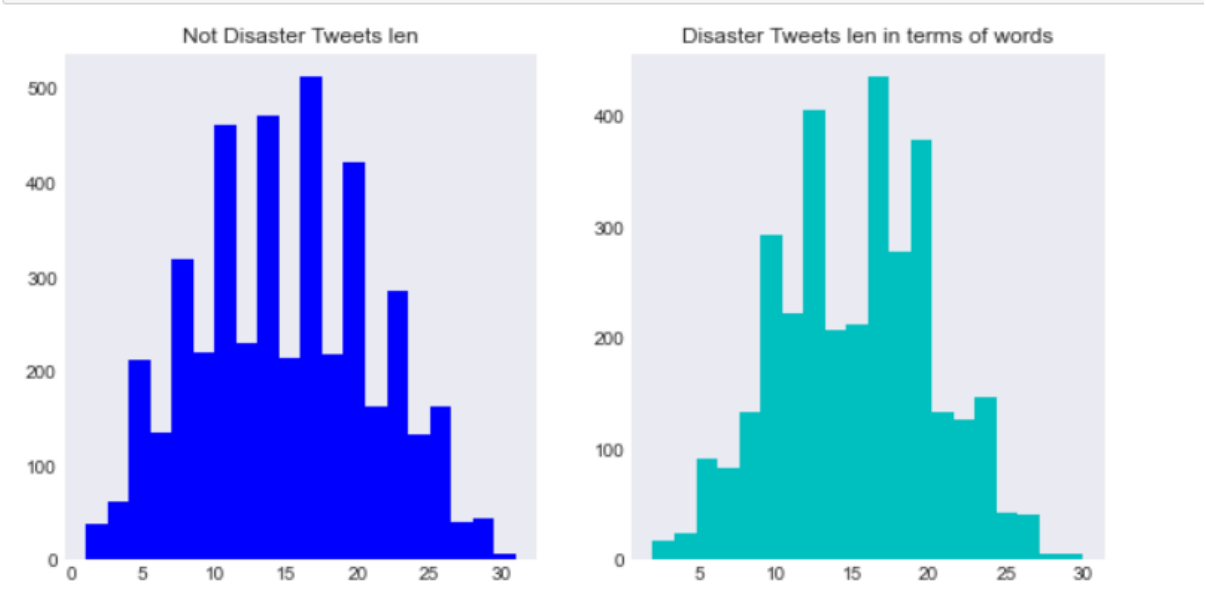
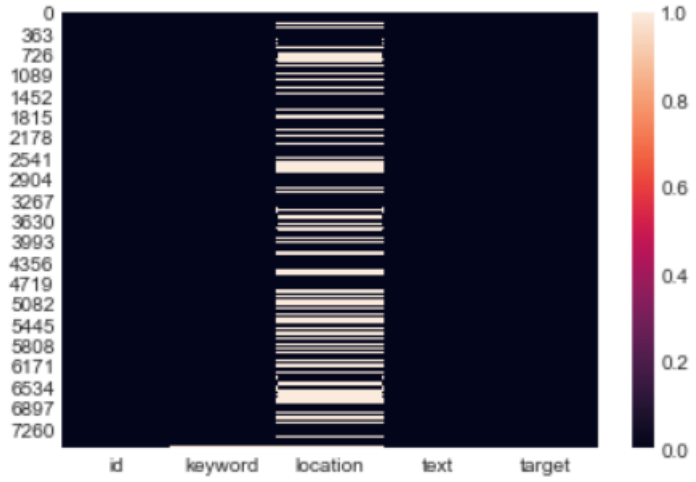
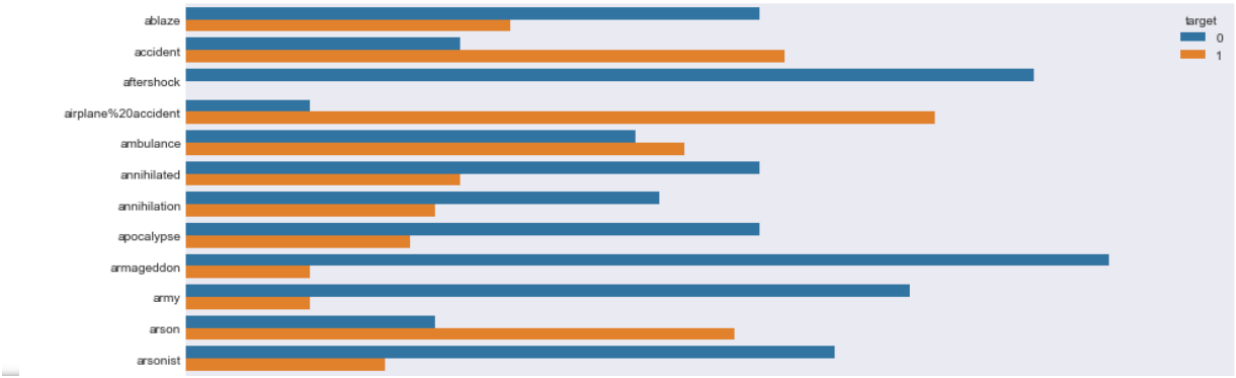


Fig-2 Tweet in terms of word length(train dataset)

Here we can see the length of tweet in term of words are more evenly distribution. Maximum number of word is tweet id around 30. There are about 500 tweets with 17 words in it.

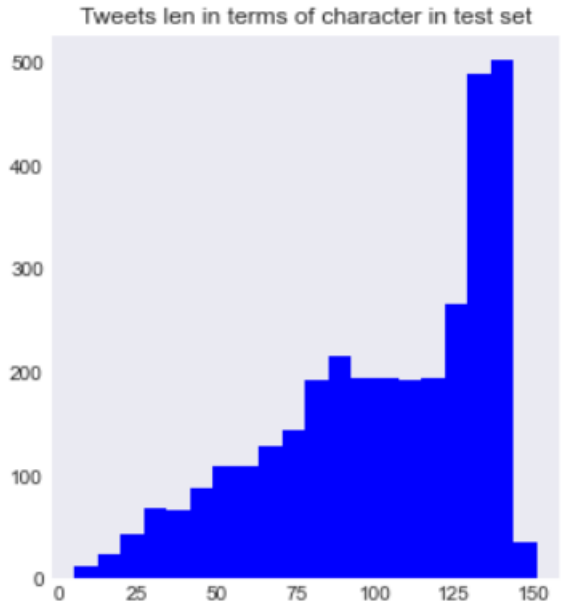


The above image is about the heat map of the train dataset the place where it is white the things are missing and from the above image we can see that many locations are missing in the dataset.



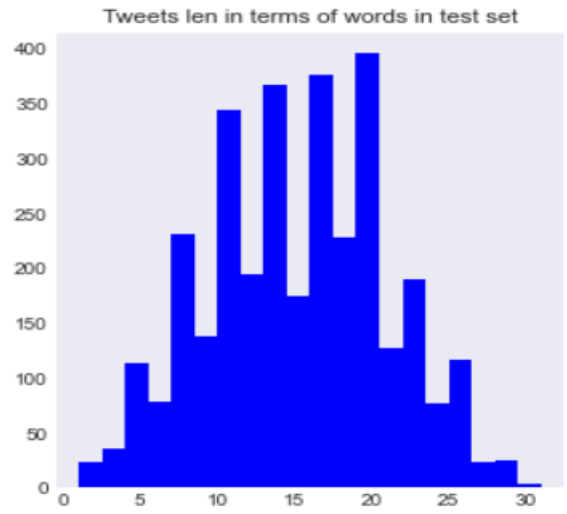
Above image shows number of tweet made with specific keyword and as we can see in some keywords non-disastrous rather than disastrous tweet. Which is one of the important part of this dataset.

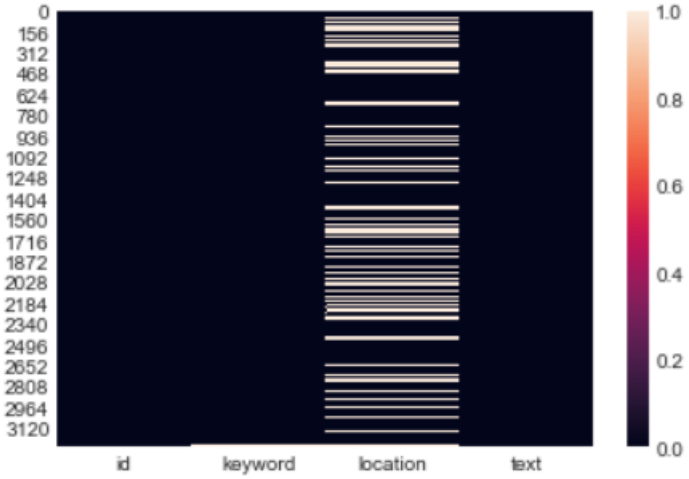
Talking about test dataset there are only four column in it, all column are same except ‘target column’ and primary goal of the project is to determine target column for this. There are 3263 data points to test the model built. Here also observations are made as train dataset.



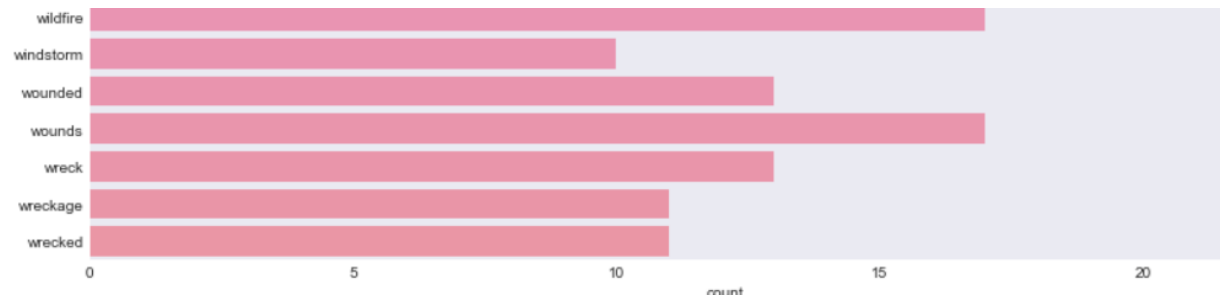
Above images gives information about tweet length in terms of characters and here also we can see similar pattern as train dataset. Number of tweets with more character is on the higher side.

Similarly length of tweet in terms of number of words and its distribution can be seen below. Most of the tweet are in range of 10-20 as shown in figure. There are maximum number of tweet with 19 words which is around 400 tweets





Above image demonstrate heat map of test dataset. Same trend is seen in test dataset as train dataset as many location places are missing. In some data points keywords are also missing.



Frequency of keywords in test dataset

PRE-PROCESSING:

One cannot train the model on a given data as it is. Here pre-processing technique play an important role. If no pre-processing is done than model will learn the features which does not lead us to a good prediction. Here as our dataset is in form of text that's why some technique required for text cleaning is mentioned below.

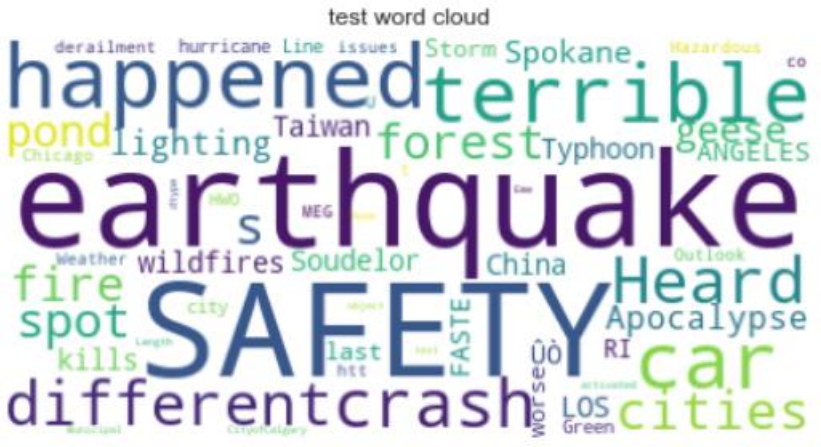
For baseline model, dropped the NA dataset and also implemented simple text cleaning with the help of nltk library which is used to import stopwords and many things but in baseline model we used only stopwords to do the cleaning.

Here dropping NA points was mistake as there are only few data points to train the model and we cannot afford to drop more.

For the final model cleaning text technique was as it used stopwords used previously but also removed the punctuations, html tags, URL and emoji's. As it will not help in predicting the model.

Stopwords are the common words which are used to make sense in the sentence but does not help in predicting test target. Same goes with the links used in the tweet and also in URLs.

Here we used word cloud to show some of the words in the train and dataset which is mentioned below.



Here words like earthquake, crash, wildfire etc gives us information about some disaster.

Particular focus should be given on cleaning in order to train the model accurately. Accuracy of model on great extend depends on how clean the text is.

Here stopwords are removed using nltk library and also to mention all the words were converted into lower case to remove the chance of duplicity. Also URLs were removed with the re library, for punctuations and emoji's we created a function to remove that.

Similarly one can use this function definition to add or remove particular from stopwords list. Here we have made no changes in that list.

BASELINE MODEL:

As mentioned in the presentation we have used logistic regression as our baseline model. To start with the model as mentioned above cleaning plays an important part in it but we can't just give text as input so changing the text to some numbers. For doing so we have used tokenizer to convert text to some sort of integer.

We have split the train dataset into two parts as train and validation set as 75% and 25% respectively. We got accuracy around 75%.

```
7]: print("accuracy :",  metrics.accuracy_score(predicted, y_test))
    accuracy : 0.75748031496063
```

Lets talk about the logistic regression in detail, logistic regression takes the word and predicts its probability. For example word earthquake will have high probability to be in the disaster column and work god will have very low probability while word crash has 50% probability to be in disaster column or not disaster column. Logistic regression dose the same and predicts the probability and if the probability is over 50% it will mark as a disaster and less than that as non-disaster. Logistic regression is imported from sikiet learn library and the predictions are made by model. In this model we can not change the accuracy as we are not selecting epochs and learning rate. In other words it is a rigid model.

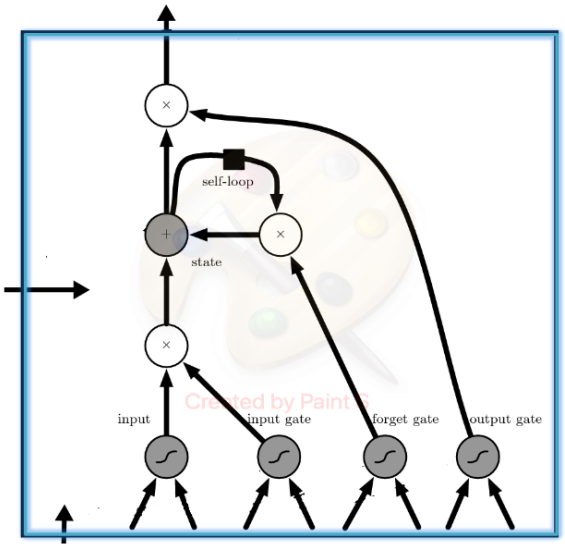
So here no further improvement was possible in the model so we tried some RNN in the next satge.

MODEL IMPROVEMENTS:

So to start with the model improvement first we changed cleaning method as in the baseline model we were only cleaning the text with the help of stop words, but in this model used stop words also punctuations, html tags and URLs and also transforming all the words to lower case. Cleaning process was already declared in the pre-processing part.

Also one thing done here is to convert works like aren't to are not as when cleaning is done aren't punctuation will be removed and that will just be are, which changes the complete meaning of the text. We define the function for both the above mentioned parts.

Now to model building, we started with the LSTM algorithm which in part of RNN and which depends on the recurrence of the event and that's how this type of algorithm works. It is a self-loop and also have outer-recurrence. Each cell has the same input and output as an RNN , but has more parameters and elements to control the flow of information.



- Idea: create self-loops to produce paths where the gradients can flow for long duration

- Input gate:
- Forget gate:
- Output gate:

Above image gives the information about the LSTM cell and there are many of such for the outer recurrence. There with the help of input, output and forget gate information is kept for the long time so that model can learn more about the specific information.

In the model first hidden layer is LSTM with 128 neuron in it which does the feature extraction part, second hidden layer was is of 128 neurons with RELU as an activation function and output layer is single neuron with sigmoid activation function to give output as ‘1’ or ‘0’.

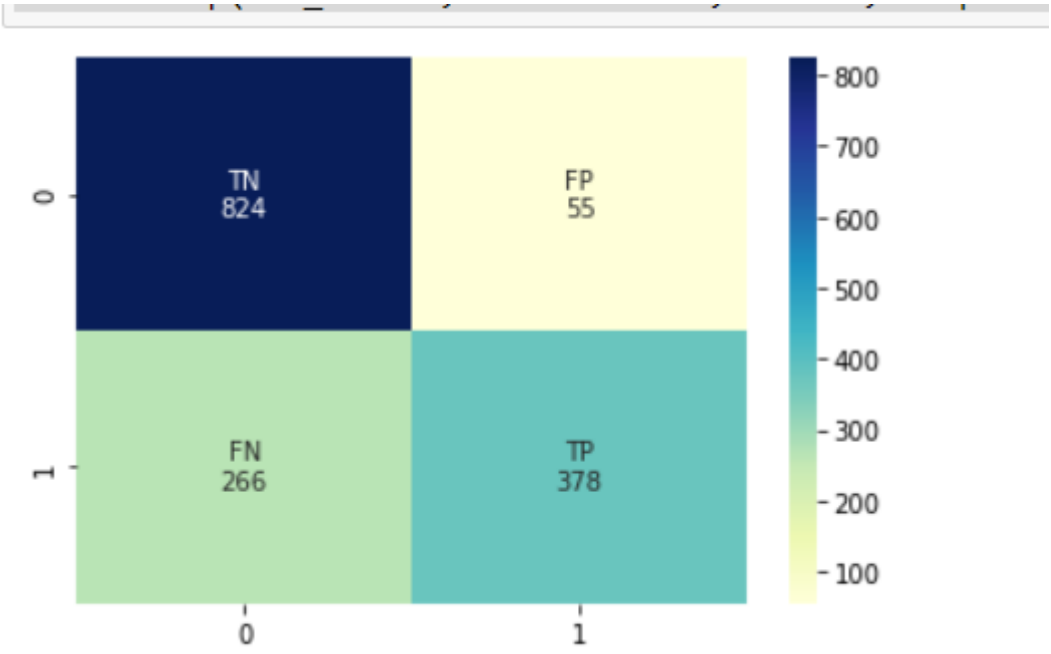
We started the training of model with 5 epochs as the accuracy for that is shown below.

	precision	recall	f1-score	support
0	0.79	0.77	0.78	879
1	0.70	0.73	0.71	644
accuracy			0.75	1523
macro avg	0.74	0.75	0.75	1523
weighted avg	0.75	0.75	0.75	1523

So here accuracy is around 75% which is near about what we got in logistic regression.

By changing the number of epoch we observed for many models for ex:- For 10 epochs accuracy was around 76%,15 epoch got same 76% but for 3 epochs we got 78% accuracy which was good but when we took deep look into it we fond the following thing.

```
Train on 5481 samples, validate on 609 samples
Epoch 1/3
5481/5481 - 17s - loss: 0.6776 - accuracy: 0.5676 - val_loss: 0.6535 - val_accuracy: 0.5731
Epoch 2/3
5481/5481 - 11s - loss: 0.6105 - accuracy: 0.6937 - val_loss: 0.5703 - val_accuracy: 0.7373
Epoch 3/3
5481/5481 - 9s - loss: 0.4347 - accuracy: 0.8280 - val_loss: 0.4635 - val_accuracy: 0.7800
```



Above images gives the information about the problem in using 3 epochs as it is performing well for disaster tweet but performing extremely on non-disaster tweets. That was the reason we choose to dump the LSTM and move towards another model.

For last model we have used glove embedding in addition to above mentioned LSTM. Now the question here is why to do so. The simple answer to this is to make proper meaning of words. As mentioned before we know that we have to tokenize the text before giving it to the model and it done in the foam of vectors. Here with the help of embedding we made the vector which was actually making the better sense of the words which it represents.

GloVe is the short form of global vectors and it uses both global and local statistics to create a word vector which is in the sense is great as if we use only local statistics it will only be good in analogy task and not in other task.

Here just simple change in the vectors made a big change I accuracy which is discussed on conclusion part.

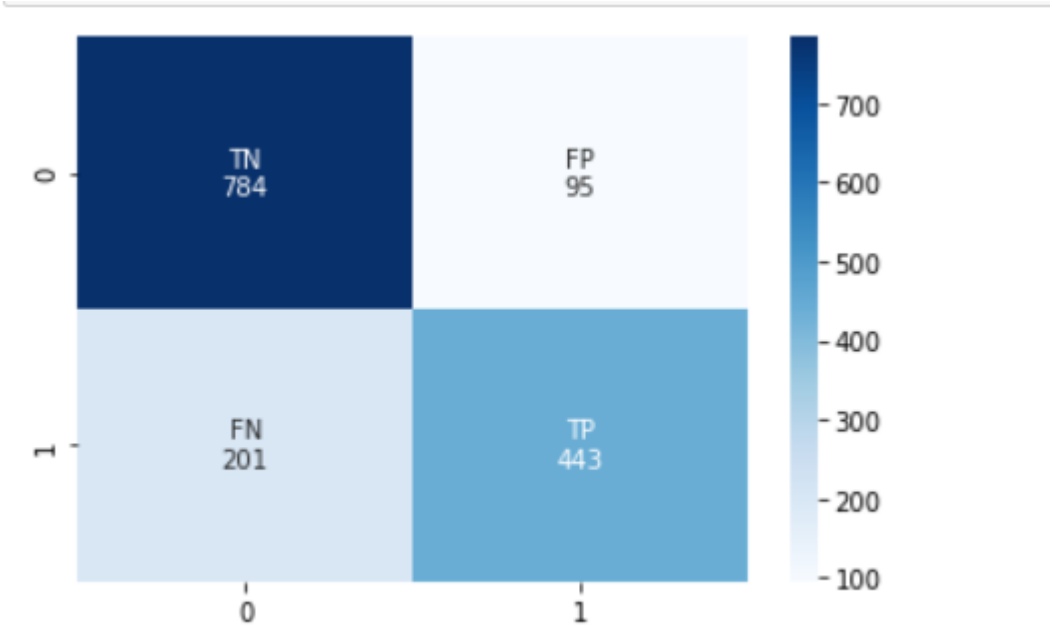
Discussion and conclusion:

Following were the results we got from our final model

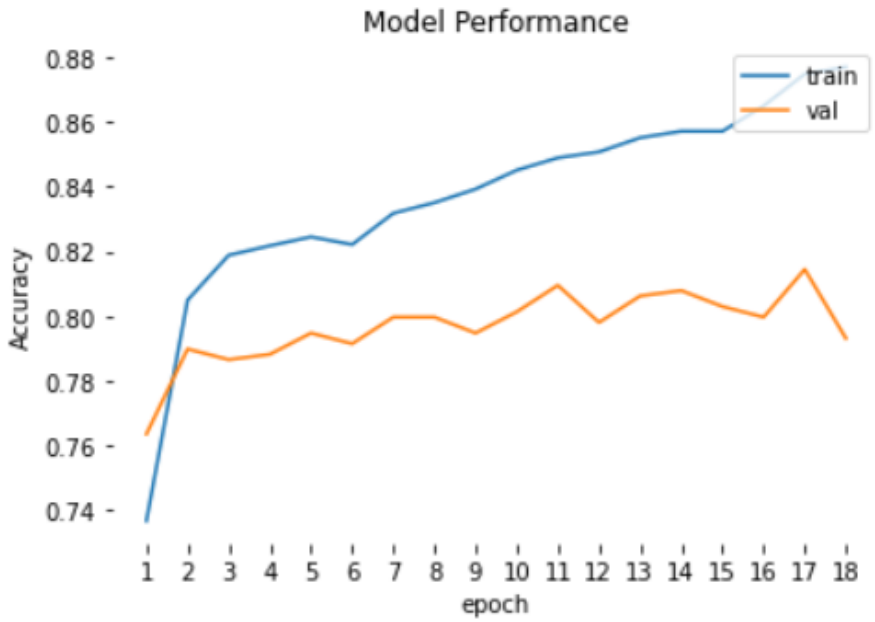
	precision	recall	f1-score	support
0	0.80	0.89	0.84	879
1	0.82	0.69	0.75	644
accuracy			0.81	1523
macro avg	0.81	0.79	0.80	1523
weighted avg	0.81	0.81	0.80	1523

Talking about the accuracy we got accuracy around 81 and which was a jump of 6% from our previous model before the embedding was done with same number of epochs.

F-1 value for disaster tweets is high compared to non-disaster tweets. And the reason is shown below



Model is performing very well on disaster tweets where as not so well on non-disaster tweets.



Here we can see that train accuracy is around 88% but validation accuracy which is the accuracy on the unseen data is around 80%. Overall the model gave accuracy around 80% on the test data.

Now taking about the area of model improvement one can still do better cleaning of the text as still as we can see in the word cloud there are many words which makes no sense in terms of model prediction. Cleaning can also be done with each section of keywords i.e. when the keyword changes cleaning method can be changed.

Also tokenization of the words can also be done in different method for example in place of GloVe one can use word2Vec to find more accurate vectors for the word.

Also Implementing the more modern NLP algorithms can help in model improvement for example BERT from google which is consider to be more accurate than any recent model.

Reference:-

1. Lecture note 12 by professor Birsen Sirkeci

2. https://www.kaggle.com/janvichokshi/disaster-tweets-cleaning-lstm-and-embedding

3. https://www.kaggle.com/danielwillgeorge/glove6b100dtxt?select=glove.6B.100d.txt

4. https://www.kaggle.com/c/nlp-getting-started