San José State
UNIVERSITY

Real or Not ? NLP With disaster tweets

Team Project – CMPE 256

Instructor : KaiKaiLiu

Summer,2020

Team 13

Team Members

Name	**Email**	**SJSU ID**
Sowmya jupally	Sowmya.jupally@sjsu.edu	014608412
Thanmai Gajam	thanmai.gajam@sjsu.edu	014604096
Laxmi Prasanna Palle	laxmiprasanna.palle@sjsu.edu	014601535

Abstract: This Project presents a system and different algorithms to classify a tweet based on the key words present. There are two main classifications i.e., disaster tweet and a non-disaster tweet. We propose an annotation schema for identifying relevant tweets as well as the more fine-grained categories they represent and develop feature-rich classifiers for relevance and fine-grained categorization.

Introduction: Social media provides a powerful lens for identifying people's behavior, decision-making, and information sources before, during, and after wide scope events, such as natural disasters. This information is important for identifying what information is propagated through which channels, and what actions and decisions people pursue. However, so much information is generated from social media services like Twitter that filtering of noise becomes necessary.

Hence in this project we focus on how we can classify tweets based on the keywords present and how to improve the efficiency in classifying them based on various algorithms and techniques. This type of automatic tweet categorization can be useful both during and after disaster events. And, we use different visualization techniques to understand the behavior of tweets along with many preprocessing techniques. The main goal will be to understand and classify the tweets based on disaster and non-disaster terms.

Related Work

1. **Data Collection:** In this section we install all the libraries necessary and then we load the data from a CSV file and run the code to understand the initial shape of the raw data.

2. **Data Preprocessing and Visualization:**

   Then the last part is to search for any missing data in any columns and those columns have been dropped. The text is cleaned further by removing all the low frequency words, unwanted URLs, emojis, html tags and punctuations to make it easier for model training. All the tweets are tokenized and finally the corpus is created using NLTK library.

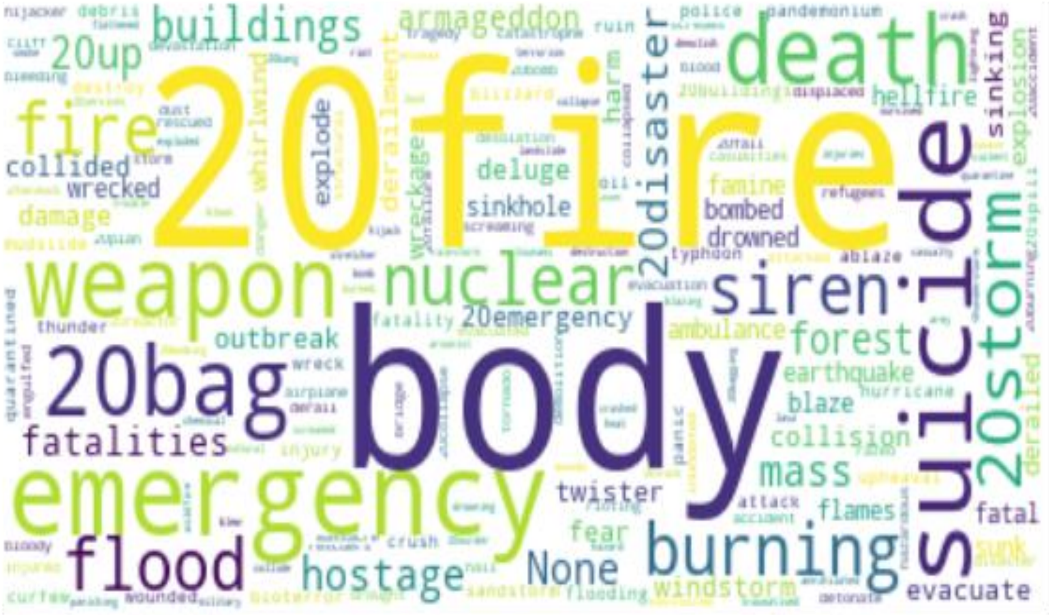
Wordcloud for Keywords:

Contributor: Laxmi Prasanna

In this we generate a word cloud by using 200 words randomly. For that word cloud generation, we used the below code on raw data

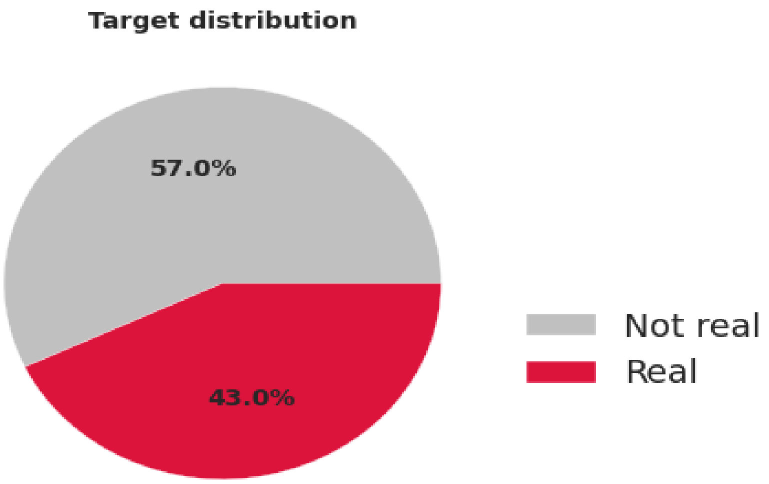
Collab Link:
Link to Word Cloud

The output image is as follows



Target distribution of real and non-real tweets:
Contributor: Laxmi Prasanna

And next we did the target distribution and get the analysis for real and not real distribution which is given as



Collab Link:

[Link to Target Distribution]

So, we have more tweets in class 0 (non-disastrous or not-real) tweet than in class 1 (Disastrous or Real) tweet.

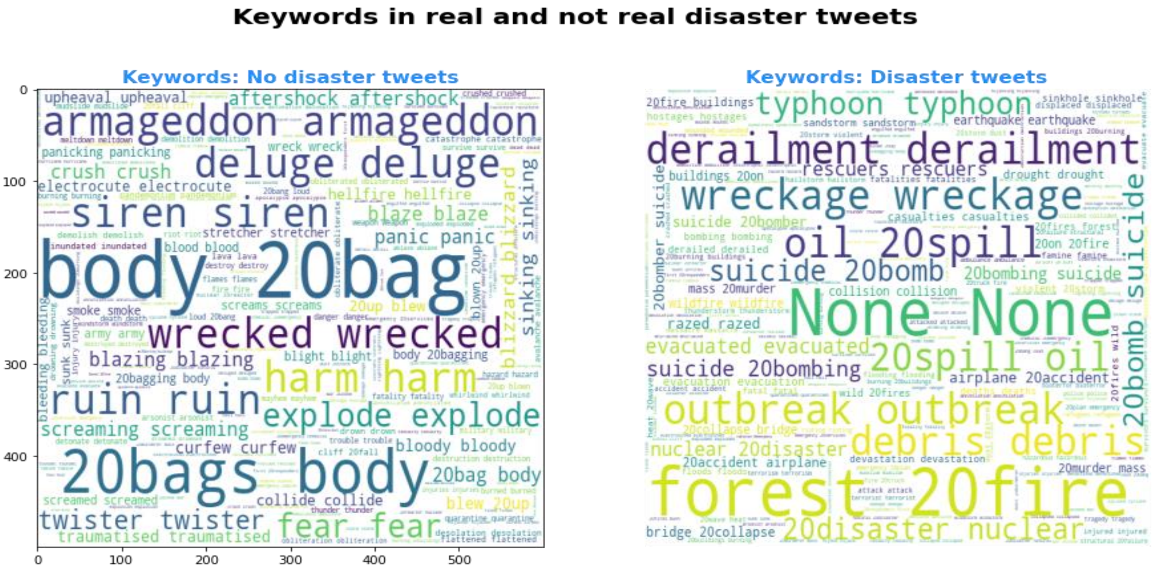
Wordclouds of keywords in real and not-real disastrous tweets

Contributor: Sowmya Jupally

And we generate a word cloud based on a disaster and non-disaster frequencies

Collab Link:

Code to Wordclouds of Keywords in Real and Not Real Disaster Tweets



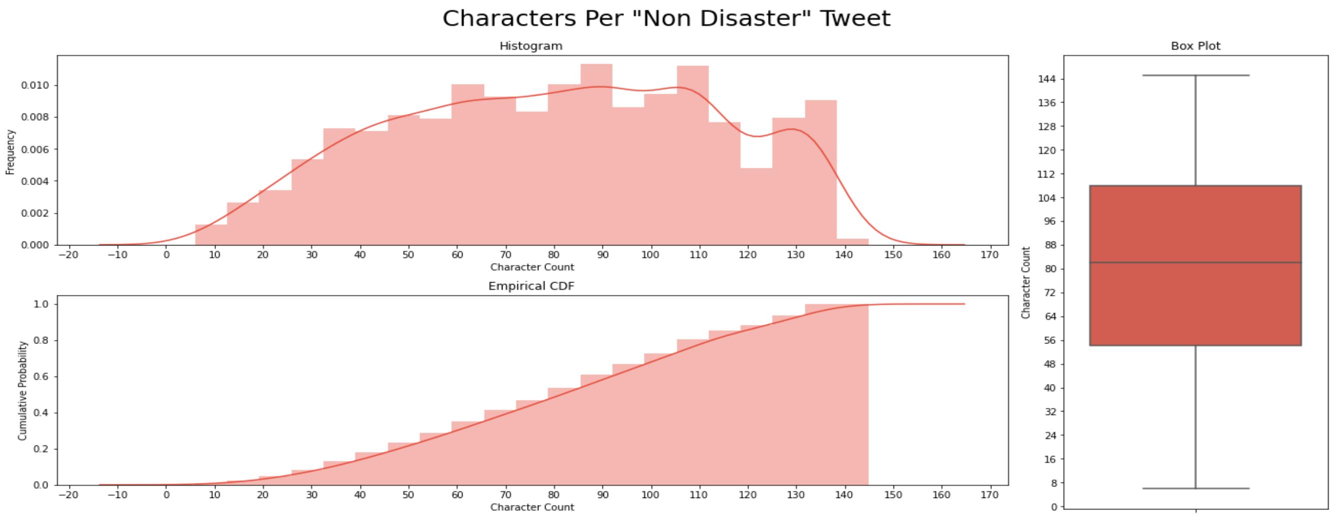
Character count distribution for real and not-real disastrous tweets:

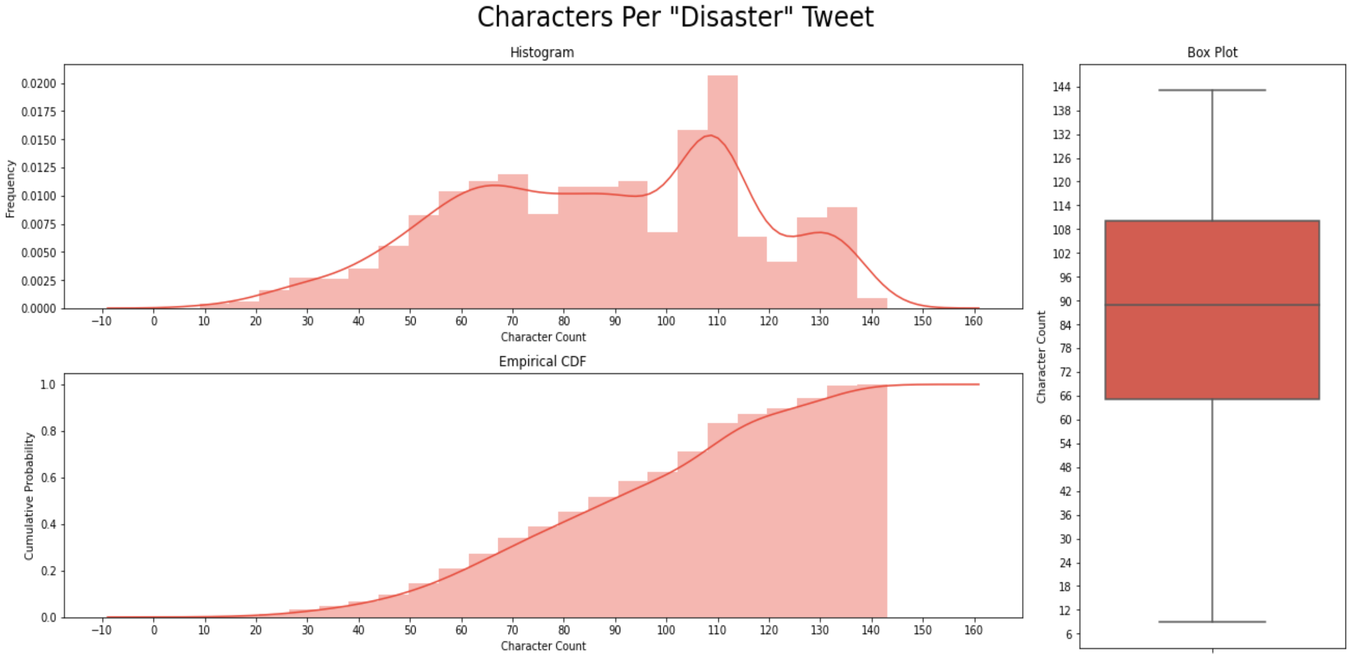
Contributor: Sowmya Jupally

We have created a new feature for visualization with already existing functions . In this based on the character count we cleaned the text for removing any html tags, smileys, symbols etc. Then a customized chart has been created for a single feature and a grid spec is used. In this we created one histogram and one empirical graph for same specification i.e., for characters length for Non-disaster tweets and disaster tweets also box plot has been shown which gives the idea of characters length that is from maximum to minimum including the average length.

Collab link:

Code to Character count distribution of real and not real disaster tweets





Word count distribution for tweets:

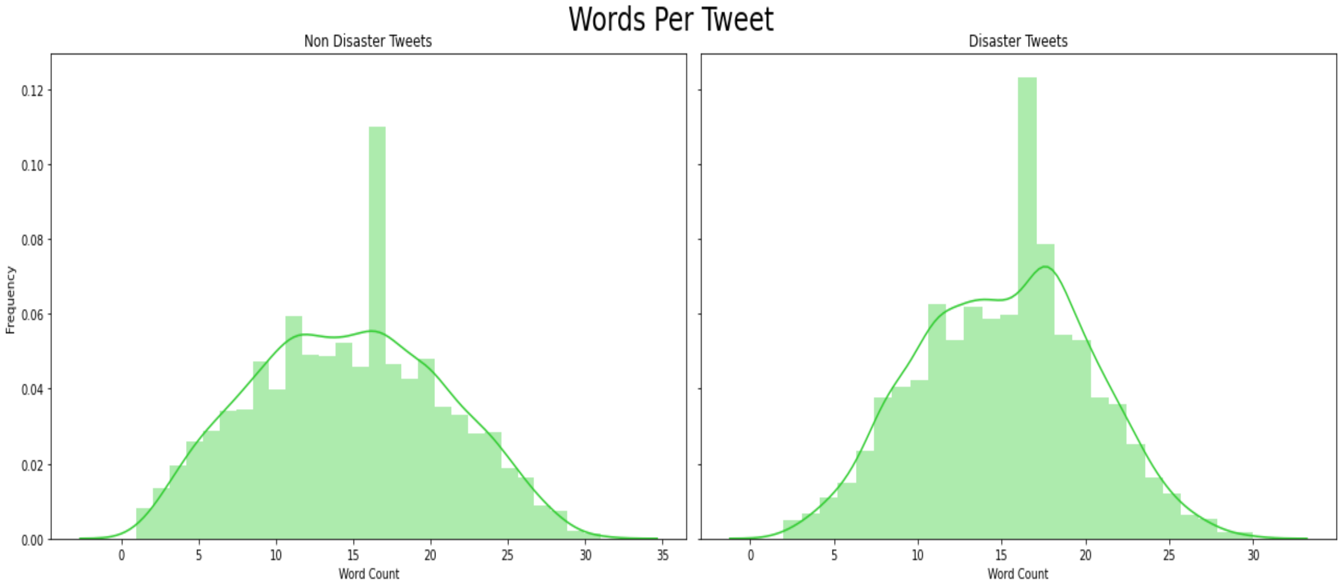
Contributor: Thanmai Gajam

And we also did word count distribution for Non disaster and disaster tweet which is a histogram plot by utilizing the function for comparing word counts. The frequency curve is also shown based on the histogram plot for understanding the frequency of the count in both tweets.

Collab Link:

[Code to Word Count Distribution for Tweets]

The histogram graph and frequency curve for word count distribution.



NLTK Pre-processing:

Contributor: Thanmai Gajam

We utilized the NLTK functionality to implement some pre-processing techniques like corpus and porter stemmer. In this section we removed we utilized preexisting functions like re.sub for removing some unwanted characters specified in the function, lower() for converting all the text into lower case letters and stem function for removing stopwords and stemming them. And after cleaning this using porter functions, we applied the cleaned text to corpus to create pandas data frame.

Also, we further cleaned the raw data using corpus and concatenated the result frames for more clear text. We created a dictionary to check the low frequency words and them we converted to dataframe to check the frequency and then we removed the words by shaping them. We also utilized bag of word techniques to create a sparse matrix to fit the corpus transformation we did above.

3. Models Building

Dataset Splitting

The dataset has a standard split, the train data set is 80% and the test data set is 20%. We give the test set to train every model.

Models Training and Evaluation:

Selection of the appropriate models is very important for having better training accuracy and validation accuracy. We here have utilized various models and fit them with various learning rates to check how the model accuracy will be. The F1 scores, train scores and test scores are provided for every model.

The models selected for this project are:

1) **Decision Tree Model:**

Contributor: Laxmi Prasanna

Decision tree model is a classification model. Decision is represented like an inverted tree, with root at it the top. The tree is split into branches and the decisions are made in the internal nodes

In this we have the entropy, and we have not utilized the max depth so that we did not change the dataset. We used best splitter along with the randomizing the dataset.

2) **Gradient Boosting Model**

Contributor: Laxmi Prasanna

Gradient Boosting model is a basic ensemble of weak prediction models. In this project we have 100 n_estimators, which represents the number of trees used in our model. The gradient boosting model also uses hyperparameters and we have included deviance loss and learning rate as 0.01 with randomized data values. By adding each weak model, the estimation further becomes accurate for the gradient boosting model.

3) **K-Nearest Neighbours Model.**

Contributor: Laxmi Prasanna

K Neighbours models uses the similarity between the data point and all the available data points which are categorized into multiple classes.

K Neighbour classifier classifies based on similarity measure. We used brute algorithms and neighbours value as 7.

4) **Logistic Regression Model**

Contributor: Sowmya Jupally

In this project, for logistic regression we have selected L2 regularization. To support L2 regularization we used saga solver which is a sag variant with randomized values. We got the best F1 score for logistic regression model.

5) **SGD Classification Model**

Contributor: Sowmya Jupally

Stochastic Gradient classifier we used hinge loss which is almost like support linear vector. And we provided model with L1 regularization with optimal learning rate.

6) **SVC Classifier**

Contributor: Sowmya Jupally

In Support vector classifier, we an select any kind of kernel, in this project we used linear kernel with default degree as 3 which means degree of polynomial kernel function. We also have classifier value as 2 which is to overfit and underfit the model.

7) Bernoulli

Contributor: Thanmai Gajam

We gave the alpha, learning rate as 0.1 and trained the model

8) Gaussian Model

Contributor: Thanmai Gajam

We used the gaussian train function and trained the model.

9) Multinomial Model

Contributor: Thanmai Gajam

We gave the alpha, learning rate as 0.1 and trained the model

10) Voting Classifier

Contributor: Thanmai Gajam

For voting classifier, we give the multiple models as input and it tells us the best fit model for the train data.

Evaluation: To find the model for the project, model evaluation is the crucial process in building a machine learning model. The test data will be given as input to the model to evaluate its performance.

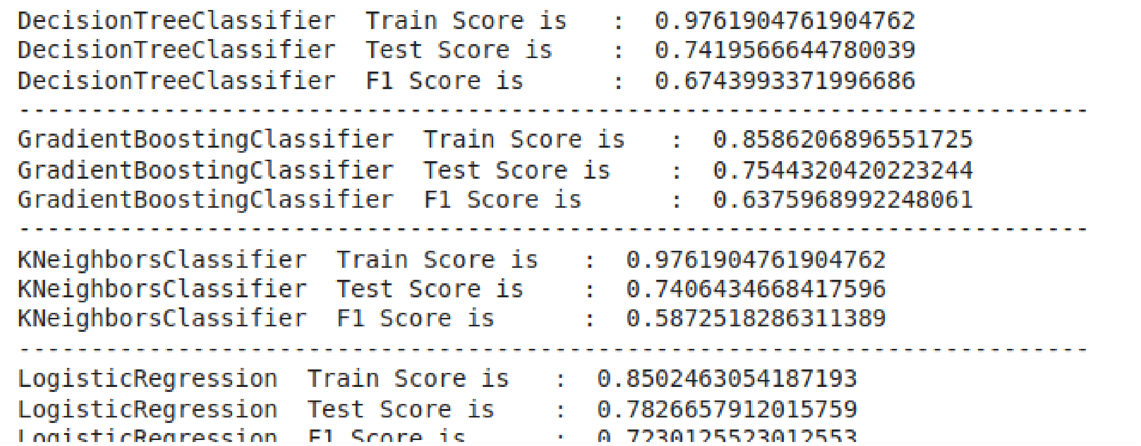
Code to the Evaluation Results

From below evaluation results we can tell that Voting classifier model and logistic regression model has highest F1 score, test score, Whereas Decision tree and K- nearest Neighbors classifier model has highest test score.

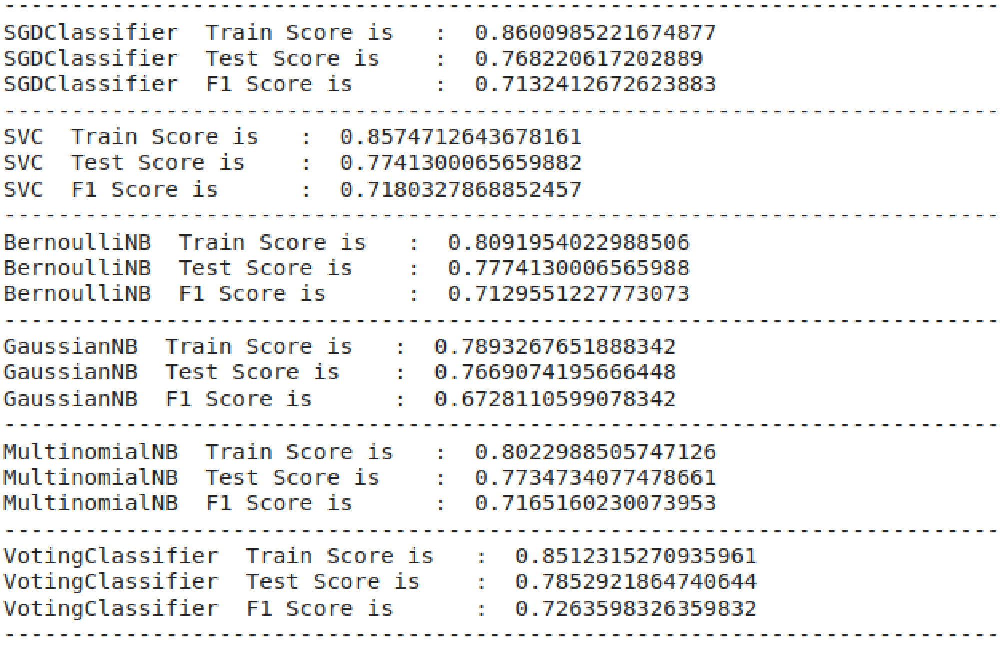
Confusion Matrix: Confusion matrix displays the False Positives, True Positives, False Negatives and True Negatives. This represents the accuracy of the model.
The proportions of True Positives and True Negatives should be closer to one, and the proportion of false positives and false negatives should be close to zero to be certain that the model is accurate.

Code Link to Confusion Matrix

The evaluation results for all models are shown below



```
DecisionTreeClassifier  Train Score is   :  0.9761904761904762
DecisionTreeClassifier  Test Score is    :  0.7419566644780039
DecisionTreeClassifier  F1 Score is      :  0.6743993371996686
-------------------------------------------------------------------------
GradientBoostingClassifier  Train Score is   :  0.8586206896551725
GradientBoostingClassifier  Test Score is    :  0.7544320420223244
GradientBoostingClassifier  F1 Score is      :  0.6375968992248061
-------------------------------------------------------------------------
KNeighborsClassifier  Train Score is   :  0.9761904761904762
KNeighborsClassifier  Test Score is    :  0.7406434668417596
KNeighborsClassifier  F1 Score is      :  0.5872518286311389
-------------------------------------------------------------------------
LogisticRegression  Train Score is   :  0.8502463054187193
LogisticRegression  Test Score is    :  0.7826657912015759
LogisticRegression  F1 Score is      :  0.7230125523012553
```



```
--------------------------------------------------------------------------
SGDClassifier  Train Score is   :  0.8600985221674877
SGDClassifier  Test Score is    :  0.768220617202889
SGDClassifier  F1 Score is      :  0.7132412672623883
--------------------------------------------------------------------------
SVC  Train Score is   :  0.8574712643678161
SVC  Test Score is    :  0.7741300065659882
SVC  F1 Score is      :  0.7180327868852457
--------------------------------------------------------------------------
BernoulliNB  Train Score is   :  0.8091954022988506
BernoulliNB  Test Score is    :  0.7774130006565988
BernoulliNB  F1 Score is      :  0.7129551227773073
--------------------------------------------------------------------------
GaussianNB  Train Score is   :  0.7893267651888342
GaussianNB  Test Score is    :  0.7669074195666448
GaussianNB  F1 Score is      :  0.6728110599078342
--------------------------------------------------------------------------
MultinomialNB  Train Score is   :  0.8022988505747126
MultinomialNB  Test Score is    :  0.7734734077478661
MultinomialNB  F1 Score is      :  0.7165160230073953
--------------------------------------------------------------------------
VotingClassifier  Train Score is   :  0.8512315270935961
VotingClassifier  Test Score is    :  0.7852921864740644
VotingClassifier  F1 Score is      :  0.7263598326359832
--------------------------------------------------------------------------
```

Convolution Neural Network Model:

Contributors: Laxmi Prasanna, Sowmya Jupally, Thanmai Gajam

Input: The model takes a batch of sentences in form of a 1-D tensor strings.
Preprocessing: It preprocesses its input by splitting on spaces.
Module also maps all out of vocabulary tokens into one bucked which is initialized with zeroes. And word embeddings are combined into sentence embeddings using the function sqrtn combiner.
Reference : https://www.tensorflow.org/api_docs/python/tf/nn/embedding_lookup_sparse

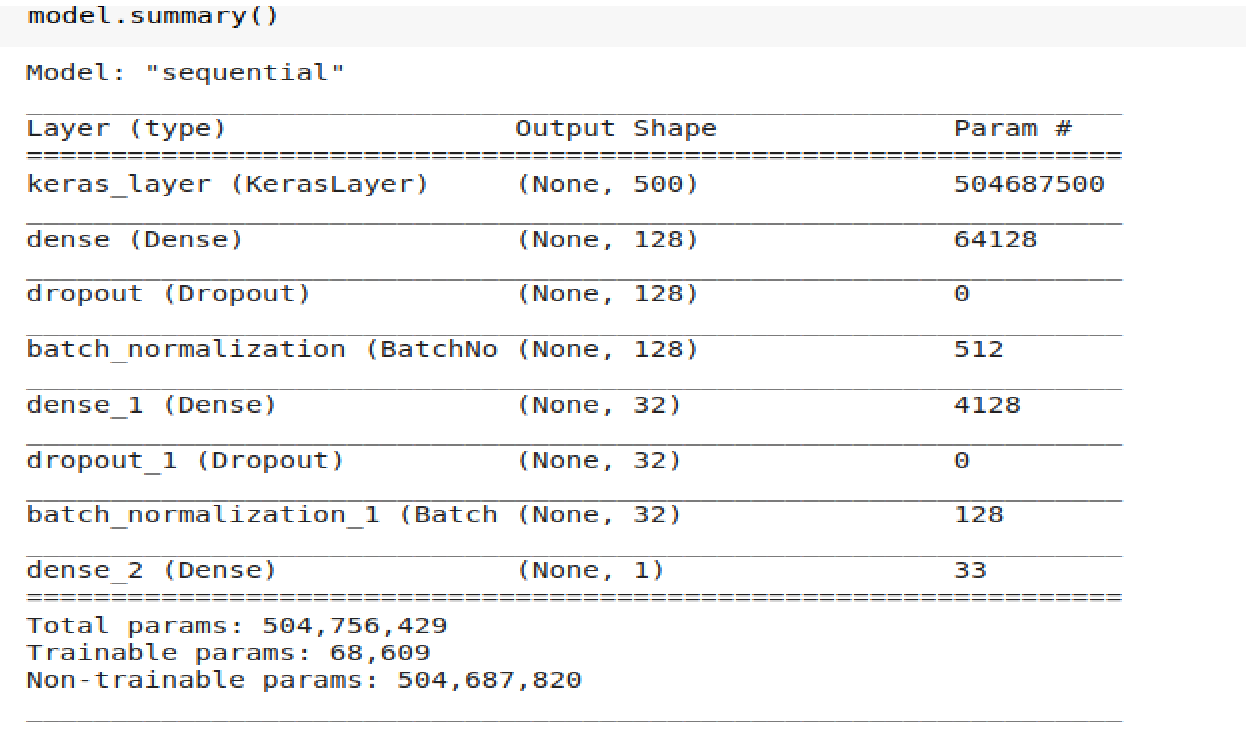
CNN Model parameters

- Dropout: We also used dropout regularization to train our CNN. We define a neural network with a dropout layer where probability = 0.4 which means less than half nodes are dropped.
- Activation function: The rectified linear activation is used for hidden layers and sigmoid activation function is used for binary classification of real and not-real tweets.
- Dense Layers: Fully connected hidden layers to perform classification.
- Output: Identifying tweet class which is real or not real i.e., disastrous, non-disastrous.
- Batch Normalization: It is used to increase the stability of neural network. This normalizes the output of previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. It reduces overfitting because it shows regularization effects on the model. Also, it adds some noise to each hidden layer activations like dropout.

Collab Link :
Link to Code

For activation code:
Link to Activation Code

The model summary is hence given as



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
keras_layer (KerasLayer)     (None, 500)               504687500 
_________________________________________________________________
dense (Dense)                (None, 128)               64128     
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
batch_normalization (BatchNo (None, 128)               512       
_________________________________________________________________
dense_1 (Dense)              (None, 32)                4128      
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 32)                128       
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33        
=================================================================
Total params: 504,756,429
Trainable params: 68,609
Non-trainable params: 504,687,820
_________________________________________________________________
```

Optimization Techniques: We use adaptive learning rate algorithm that's been designed specifically for training deep neural networks. And we set the learning rate to 0.0002.

Collab Link:

Link to Optimization Techniques

Callbacks List:

The Optimization technique called early stopping is implemented for CNN model to reduce overfitting while training the model. Early stopping terminates the training of neural network model at an epoch before it becomes over-fit. The epochs are set to 5. Initial callback early stopping is to early stop the method, the second callback checkpoints is to save the current model. And we check the model and save only when its best.

Collab Link:

Link to Callbacks Code

And we train the model with epochs = 30.

The loss accuracy and epochs are then calculated. The link for epochs code and results:

Link to Code

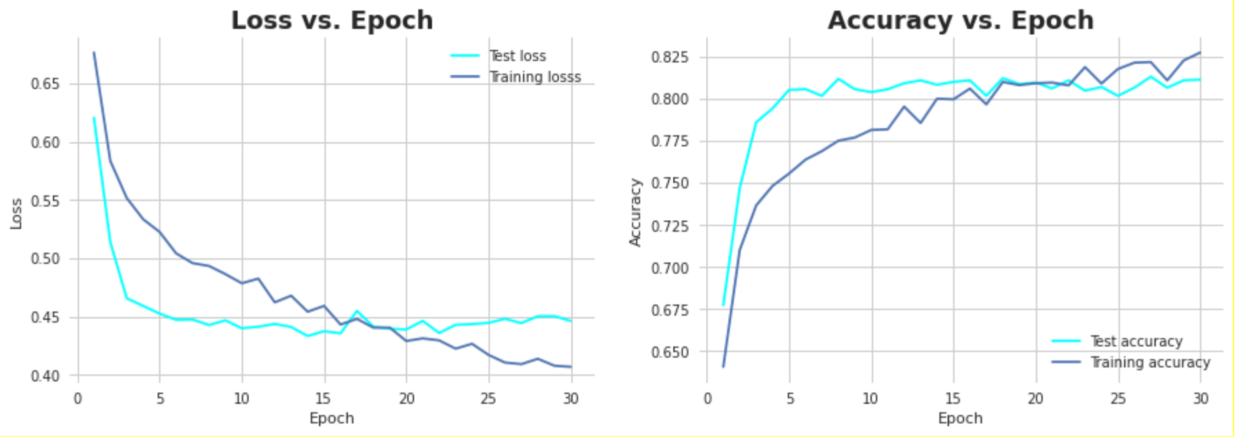
7. Model Evaluation

To find the model for the project, model evaluation is the crucial process in building a machine learning model. The test data will be given as input to the model to evaluate its performance.

The loss vs epoch and accuracy vs epoch graphs are displayed below.

Collab Link:

Link to Evaluation Code



Prediction: Using the test data we predict the values.

Collab Link:

Link to Code

References:

- https://www.aclweb.org/anthology/D11-1145.pdf
- https://medium.com/real-or-not-nlp-with-disaster-tweets/real-or-not-nlp-with-disaster-tweets-a-data-science-capstone-project-fafa6c35c16f
- https://www.kaggle.com/c/nlp-getting-started
- KaiKaiLu – Lectures and collab notes